GEORGIA INSTITUTE OF TECHNOLOGY LIBRARY

Regulations for the Use of Theses

Unpublished theses submitted for the Master's and Doctor's degrees and deposited in the Georgia Institute of Technology Library are open for inspection and consultation, but must be used with due regard for the rights of the authors. Passages may be copied only with permission of the authors, and proper credit must be given in subsequent written or published work. Extensive copying or publication of the thesis in whole or in part requires the consent of the Dean of the Graduate Division of the Georgia Institute of Technology.

This thesis by JOHN DANIEL ESKEW has been used by the following persons, whose signatures attest their acceptance of the above restrictions.

A library which borrows this thesis for use by its patrons is expected to secure the signature of each user.

NAME AND ADDRESS OF USER	BORROWING LIBRARY	DATE

ECONOMIC ASPECTS OF INDUSTRIAL SCHEDULING

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

by

John Daniel Eskew

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Industrial Engineering

Georgia Institute of Technology

November, 1973

ECONOMIC ASPECTS OF INDUSTRIAL SCHEDULING

APPROVED:

R. G. Parker, Chairman

R. H. Deane

D. C. Montgomery

Date approved by Chairman: 4/20/73

ACKNOWLEDGEMENTS

My greatest appreciation goes to Dr. Robert G. Parker, who throughout this research has provided the guidance to make its completion possible. Dr. Parker's willingness to turn his attention to this work whenever I needed his assistance will not be easily forgotten.

Professors Richard H. Deane and Douglas C. Montgomery deserve special thanks for their helpful suggestions. Thanks are due Dr. Russel G. Heikes who helped with the thesis presentation on short notice. I am very grateful for the aid of Elise Ashley and Valorie Hall in preparing the final copy.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF ILLUSTRATIONS

SUMMARY

The operations research literature abounds with various approaches to the job shop scheduling problem. These approaches have not only failed to yield efficient solution procedures for problems of reasonable size, but have also failed to provide insight into the economic aspects of scheduling. The purpose of this research is to explore these economic aspects from the standpoint of identifying costs involved, investigating characteristics of schedules in the economic realm, providing solution procedures which yield good schedules with respect to economic measures, and investigating the sensitivity of these solution procedures to the various cost parameters.

With this information at hand a heuristic algorithm is developed to minimize the sum of machine idle cost and job penalty cost. A branch and bound technique is used, then additional consideration is given to job waiting cost after the first pass solution is obtained. By use of computer testing, this algorithm is compared with a schedule time algorithm and a dispatching rule. Sensitivity tests are then conducted to determine the effects of variation of selected parameters on the analysis.

It is found that the cost oriented algorithm offers much improved solutions in the economic realm when compared with more conventional procedures. These results are, however, parameter dependent. Significant recommendations for further study are given.

CHAPTER I

INTRODUCTION

In the years since World War II a great deal of progress has been made in the application of mathematical techniques to solving complex problems in business and industry. Management scientists have been successful in increasing profits through better business planning and control in uncountable situations. One area of interest to the industrial manager which has not sufficiently yielded to the methods of the management scientist is the job shop scheduling problem. This research approaches the shop scheduling problem from the standpoint of maximizing profits, or minimizing costs, rather than the more usual approach of considering the problem from the standpoint of measures which are functions of time only, thus matching the criterion of optimization with the real world objective.

The Shop Scheduling Problem

The shop scheduling problem is the problem of scheduling J jobs on M facilities, or machines, where each job is composed of several operations constrained to be accomplished on specified machines in a prescribed sequence. The objective is to sequence the operations on each machine so as to extremize some measure of performance. It is normally assumed that a given operation can be performed on only one of the machines; hence, the sequences of the required machines for each job are known, and the sequences of the jobs on the various machines

are to be determined. A schedule is known only after this sequencing is accomplished and clock times have been associated with the various events. The principal difficulty in solving this problem is the size of the possible solution set, since there are $(J!)^M$ possible sequences for the case where each job is processed once and only once on each machine, each of these sequences having an unlimited number of possible schedules. Even though the nomenclature of the shop scheduling problem has historically been associated with that of the machine shop, the structure of the problem is such that it should not be construed that the production machine shop is the only area of application.

The shop scheduling problem can be classified in various ways, two of which are the nature of job routing and the nature of job arrivals. If each job has the same machine ordering restrictions a flow shop is said to exist. In this case the machines can, without loss of generality, be numbered 1, 2, 3, . . . , M-1, M, and each job will be processed on the machines in that order. If all jobs are not constrained to have the same machine ordering a job shop is said to exist. If at the time of scheduling, all jobs together with their arrival times at the shop are known, the problem is said to be static, but if jobs can arrive at unknown times in the future the problem is dynamic. Unless stated otherwise the current work will deal with the static case.

Historically the problem solver has concerned himself with scheduling to minimize total elapsed time to complete all jobs, usually referred to as the make-span or schedule time. It will be seen that this is not necessarily consistent with the manager's usual goal of maximizing profit. First, however, more background information is

presented in order to intelligently discuss the scheduling problem. The reader desiring more depth than that given herein is advised to consult other more comprehensive works [3, 17].

Assumptions

Many variations in the job shop structure can arise. For example, a certain machine may be able to process two jobs at once, or a shop may have multiple facilities capable of processing similar jobs. In order to provide a uniform basis for attacking the problem and to facilitate solution procedures, the following simplifying assumptions are made:

1. No job can be in process on more than one machine at a given time.

2. No machine can process more than one job at a given time.

3. Once started on a machine, an operation must be processed until completion.

4. Processing times of operations on machines are known and independent of sequence.

5. A given operation can be performed by only one machine in the shop.

6. Machines are continuously available. The only conflicts to be resolved are among the operations themselves.

7. In-process inventory (job waiting) is permitted.

Rarely will a situation occur where all of these assumptions hold strictly. As an example, it would not be unusual for the jobs to be job lots, so that as soon as one unit of the lot is completed it is sent to the next machine, in violation of assumption (1). In such a

case, two approaches are possible:

1. Solve the problem using a procedure in strict compliance with the assumptions as listed and be satisfied with the resulting suboptimal solution.

2. Develop a solution procedure which allows the assumption to be relaxed. It is important to realize that even though certain assumptions must be made to make the general problem tangible, the assumptions must be carefully weighed for applicability when used in a real world situation.

Measures of Performance

Central to the topic of this work will be the performance criteria by which the shop schedule is measured. Mellor [51] lists several properties of desirable schedules as taken from other works. Some measures of performance which might be of concern are:

1. Time to complete all jobs, frequently referred to as make-span or schedule time.

2. Machine utilization, or the percentage of time the machines are in use for processing jobs.

3. Number of tardy jobs.

4. Cost of tardy jobs.

5. In-process inventory costs.

6. Cost of processing jobs.

7. Cost of idle machines.

8. Workload balance among machines.

9. Mean number of jobs in the shop.

10. Mean lateness.

11. Lateness variance.

12. Mean time in the shop.

The above list is not intended to be exhaustive. There are relationships between many of these measures of performance. Minimization of schedule time for a given set of jobs corresponds to maximization of average machine utilization, but does not necessarily correspond to minimization of costs of tardy jobs since due dates and penalties for being tardy must be considered. There are some industrial situations wherein consideration of only one of these measures is adequate. For example, some industries operate with a large portion of their capital invested in in-process inventory. In this situation the best criterion for production scheduling might be minimization of in-process inventory cost. But, as Gupta [33] has shown, no one of these measures of performance can be used as the criterion to yield a minimum cost solution to the scheduling problem in the general case. After a review of the literature cost considerations will be discussed in more detail.

Review of the Literature

Research into the static scheduling problem has, to date, suffered from at least two important inadequacies. First, the structure of the problem as defined is rarely compatible with the problem as it exists in the real world. Specifically, the measure of performance used has almost exclusively been schedule time, and the assumptions used have been, for the most part, those given in the previous section, even though such assumptions are not necessarily realistic. Second, even with the problem so rigidly structured, efficient solution procedures

for problems of reasonable size have not been found.

Perhaps the most important single work in the area of scheduling is Johnson's solution to the two machine flow shop problem [42], not so much because of the practical impact of the results but because of the influence the work had on the direction of future research. Most of the works cited below use the criterion adopted by Johnson - minimization of schedule time.

The literature survey presented below is, of course, not exhaustive. The bibliography contains a more comprehensive listing of pertinent works. The literature will be considered in four separate categories according to solution methodology: combinatorial methods, mathematical programming, heuristics, and other approaches.

Combinatorial Methods

Combinatorical methods rely on a systematic generation of sequences in an iterative manner. Some of the earliest approaches to the scheduling problem, as references [42] and [58], were combinatorial in nature. McNaughton [50] used a combinatorial approach to minimize lateness penalty for single machine and identical machine shops. In 1964 Dudek and Teuton [20] introduced an algorithm to minimize schedule time in a flow shop by filling the job sequence positions one at a time. At each position various decision rules were used to compare the jobs which could be sequenced next. In this way a subset of sequences was generated and the best of the subset was found by enumeration. This procedure was later shown suboptimal by Karush [43] only to be subsequently improved by Smith and Dudek [57].

Another combinatorial approach is branch and bound with back-

tracking, sometimes referred to as combinatorial programming. The branch and bound methods embody construction of the solution space as a tree, with the nodes representing selection of particular sequences. The idea is to branch to the node having the least lower bound on the measure to be optimized. Branch and bound techniques are all rather recent in development. Ignall and Schrage [41] offered algorithms giving optimal solutions to the two machine flow shop mean flow time problem as well as the three machine flow shop minimum make-span problem. Brooks and White [11] developed a branch and bound approach to the general job shop problem and offered evidence that the technique could have merit in solving problems using criteria other than minimum make-span, such as job lateness penalty. Brown and Lomnicki [12] offered an algorithm giving an optimal solution to the flow shop minimum make-span problem of only permutation sequences are considered. Most of the recent attention in branch and bound with backtracking has focused on the use of a graph theoretic representation of the problem, as in references [9], [13], and [27].

Mathematical Programming

Attempts to solve the job shop scheduling problem using conventional programming techniques have proven to be of little more than analytical verification. A dynamic programming solution was proposed by Held and Karp [38] to minimize the production cost for the one machine case and the parallel identical machine case. Several authors, such as Wagner [59] and Manne [48], have presented integer programming formulations of the job shop problem. Moreover, Manne points out that the structure of the problem fits very well into the mixed integer

programming formulation but that unfortunately algorithms of sufficient power are not available to solve the problem.

Heuristics

Heuristic techniques, usually of the branch and bound type, seem to offer the most promise of solution methods currently available. The branch and bound methods are like the branch and bound mentioned under combinatorial methods except that backtracking is not used and hence an optimal solution is not likely. The advantage offered is that much less computation time and computer memory are required. Even though any of the branch and bound with backtracking algorithms can be used as heuristic algorithms, Ashour and Parker [5, 6] have treated the heuristic case explicitly.

Other Approaches

Due to the large number of possible solutions to the scheduling problem complete enumeration is, of course, unreasonable for problems of realistic size. However, there has been some attention directed toward enumeration of a logically constructed subset of the possible schedules, as in references [30] and [39]. One work [1] proposed partitioning the set of jobs into smaller subsets, solving the subsets as separate problems by some accepted technique, and then combining solution results to obtain a schedule for the original set of jobs. None of these techniques have gained wide acceptance.

It is appropriate here to mention the dynamic scheduling problem. Since due to the random arrivals it is impossible to solve the dynamic problem in the same sense the static problem is solved, attention has been directed toward developing decision rules which offer a high

probability of yielding a good schedule when applied to the problem as it unfolds. The results of the process, sometimes called probabilistic dispatching, depend primarily on the selection discipline used to select a job for processing from among those waiting in a queue whenever a machine becomes available. Many different measures of performance have been proposed for the dynamic shop, some of which were listed earlier. Descriptions of these measures and related experiments can be found in references [14], [15], [16], [19], and [29]. In addition, LeGrande [46] has used a weighting of some of these measures and others in an attempt to evaluate various priority rules in terms of their effect on total cost in a particular shop. Holt [40] has also considered using cost measures in a dynamic shop. These studies with weighted and cost measures take on added significance with respect to the current research since the static scheduling problem sometimes arises as a portion of a much larger dynamic problem.

Proposed Research

This research is concerned with the economic aspects of shop scheduling. The relevant costs to be considered by the manager involved with the scheduling process are explored. Assumptions necessary to clearly define and mathematically formulate the cost components are stated, and some of the characteristics of schedules are analyzed.

A heuristic algorithm designed to explicitly consider economic factors in the scheduling process is developed. Computer experimentation is used to investigate the performance of the algorithm as compared to the performance of previously available methods, and to investigate the sensitivity of the algorithm to various parameters.

CHAPTER II

DEVELOPMENT OF A COST-BASED ALGORITHM

From the insight gained in Chapter I, the conclusion can be drawn that most of the research in scheduling theory has been directed toward the attainment of a solution which is optimal with respect to some measure explicit in the time domain, usually schedule time. However the manager of an enterprise in which scheduling is important is usually more concerned with the maximization of profits, or assuming constant revenue, minimization of cost. Thus the theoretician's objectives have not necessarily been consistent with real world objectives. The current chapter presents one approach by which good solutions can be obtained when the objective is to minimize the total cost due to the generated schedule.

A Case for Economic Considerations

Conway states that the principal costs affected by the scheduling decision are the costs of inventory, utilization, and lateness [17, p. 21]. Gupta [33] adds to these, the operation cost or the component of cost incurred in actual production. Gupta, as well, presents several approaches dealing with economic aspects of scheduling and concludes that minimization of total opportunity cost is the most viable since opportunity cost is an accurate measure of the cost of departure of a given schedule from an ideal schedule. The results of his experiments indicate that minimization of schedule time is a relatively poor criterion

when the goal is to minimize total cost.

The Cost Components

Opportunity costs reflect the loss of profit which results from rejecting a particular course of action available to the decision maker. These costs are difficult to measure, since by definition, the actions which would cause them to appear on the profit and loss statement may never be carried out. The cost formulations presented herein are based on assumptions which at times may not be representative of the real situation, and hence they are not intended for indiscriminate use in all shop scheduling problems. The reader must keep in mind, however, that any mathematical model allowing an algorithmic solution is an abstraction and will, necessarily, fail to represent perfectly the real world system. The following formulations for costs are adapted from those of Gupta [33] but are based on somewhat different assumptions, as explained below.

Operation Cost. Operation costs are those costs which arise due to the actual processing of a job on a machine. These costs can be assumed to be composed of the cost of running the machine in addition to the cost of setup, where the cost of running the machine is sequence independent. The only variable component is then due to sequence dependent setup costs, which will not be considered in this research. Then under the assumption that setup costs and setup times are sequence independent, the operation cost component of opportunity cost will be disregarded. The scheduler must keep in mind however, that there are occasions when this assumption is not valid and the resulting consequences must be investigated.

Job Waiting Cost. Job waiting costs, or in-process inventory costs, are those costs incurred while a job is in the shop with no work being performed on it. The costs incurred are quite similar to those incurred for holding raw materials or finished goods inventories. The largest of these is usually the return lost because the capital invested in inventory cannot be used elsewhere in the enterprise to finance profit making projects. Other costs include losses due to breakage and pilferage, taxes, insurance, and cost of providing storage space. Certainly, these may depend on how the individual firm operates its business. Quite often, for example, the storage space provided is not dependent on the scheduling policy, and in this case the cost of storage space is simply a part of factory overhead. A particular cost should be included in the opportunity cost if the scheduling policy will affect the amount of cost incurred.

Since most of the job waiting costs mentioned above are closely related with the value of the product, it is reasonable to assume that the charge for job waiting can be expressed as a fraction or percentage value representative of dollar charge per dollar held in in-process inventory per unit time that the product is not being processed. If this rate is r, and the value of the product at some time is V, then the job waiting cost per unit time is rV. In order to determine the total job waiting cost component, certain assumptions must be made concerning the time frame over which the costs are incurred. Raw materials may be considered available at any time from the start of the first operation of the job set until the start of the first operation for the job being considered. Since the former is more likely to be

the case for most manufacturing situations, it will be used herein. Likewise, in-process inventory charges continue to accrue until the job is delivered, where delivery might occur upon completion of the last operation of the job under consideration, at the due date if it is subsequent to the job completion time, or at any point in between. Here it will be assumed that the job is held until the due date or job completion, whichever is later. In his development, Gupta assumed that no waiting costs were incurred after the last operation of a job.

Let V_{il} be the cost of raw materials for job i, and v_{ij} be the value added to the product by operation j. Then V_{in}, the value of job i before operation n, is given by:

$$V_{in} = V_{il} + \sum_{j=1}^{n-1} v_{ij};$$

$$i = 1, 2, \ldots, J$$

$$n = 2, 3, \ldots, g_i + 1 \qquad (1)$$

where g_i is the number of operations in job i and $V_{i(g_i + 1)}$ is the value of job i after the last operation.* Further let y_{ij} be the time job i waits before operation j and after operation j - 1, and $y_{i(g_i + 1)}$ the waiting time from C_i, the completion time of job i, to d_i, the due date of job i, then β_i, the job waiting cost component for job i can be given such that:

* A glossary of notation used is provided in Appendix B.

$$\beta_i = \sum_{n=1}^{g_i+1} r[V_{il} + \sum_{j=1}^{n-1} \upsilon_{ij}\ ]\ y_{in}\ ;$$

$$i = 1, 2, \ldots, J \qquad (2)$$

The job waiting component, $\beta(s)$, of the total cost is then:

$$\beta(s) = \sum_{i=1}^{J} \sum_{n=1}^{g_i+1} r[V_{il} + \sum_{j=1}^{n-1} \upsilon_{ij}\ ]\ y_{in} \qquad (3)$$

It can be seen that as a job moves through the shop and accumulates value, the waiting cost per unit time for the job increases. It would appear desirable then, to add value to a product as late as possible in order to minimize job waiting cost. Such a conjecture is pursued again, subsequent to consideration of additional pertinent cost aspects.

Machine Idle Cost. As J jobs are scheduled on M machines, there will usually be some time when the various machines are idle at which point an opportunity cost should be affixed. The cost to be charged is the return lost because of the machine not being used for productive work. Hence, if in the absence of the J jobs the machines could still not be used for other productive work, the opportunity cost should be zero. Let the opportunity cost for machine k be r_k dollars per unit idle time. Although it is possible that r_k could be a function of time, time variations are not likely to be large over a given job set, and

will not be considered here. r_k is simply the difference between revenue per unit time obtained from machine k and the cost per unit time of operating machine k, not including the overhead costs which are independent of operating hours.

Implicit in the consideration of the static problem only is the assumption that batch processing is used, or that the shop processes only J jobs from time zero until T(S), the completion time of the last operation of the J jobs. Idle costs should then be charged for all machine inactive time from time zero until T(S). For a given set of jobs the total idle time for each machine is known once T(S) is known because the processing times of all the operations on each machine are given. Let I_{ijk} be the idle time on machine k before the processing of operation j of job i on machine k, and C^k be the completion time of the last operation processed on machine k. Note that $T(S) = \max_k [C^k]$. Then machine idle time for machine k is given by:

$$I_k = \sum_{i,j} I_{ijk} + T(S) - C^k = T(S) - \sum_{i,j} t_{ijk} \; ;$$

$$k = 1, 2, \ldots, M \qquad (4)$$

where t_{ijk} is the processing time of operation j of job i on machine k. Further, the machine idle opportunity cost for machine k, γ_k, and the total machine idle cost for schedule S, $\gamma(S)$, are given by:

$$\gamma_k = r_k I_k = r_k[T(S) - \sum_{i,j} t_{ijk}] ;$$

$$k = 1, 2, \ldots, M \tag{5}$$

$$\gamma(S) = \sum_{k} \gamma_k = \sum_{k=1}^{M} r_k[T(S) - \sum_{i,j} t_{ijk}] \tag{6}$$

Since processing times are constant, it is clear from the above formulation that minimization of schedule time corresponds to minimization of machine idle cost, regardless of the values of r_k. This was not the case for Gupta's formulation, since he assumed that idle costs were not incurred for machine inactive time after the processing of the last operation on each machine.

Job Penalty Cost. Of the costs arising from or bearing on the scheduling decision, job lateness costs are the most difficult to quantify because they cannot be separated from the behavioral aspects of the customer-supplier relationships. Considerations which must enter into determination of lateness costs, as suggested by Gere [29], include:

1. Contractual penalty clauses,
2. Costs of dealing with the customer,
3. Costs of expediting tardy jobs or holding early jobs,
4. Customer dissatisfaction.

Costs due to contractual penalty clauses are easy to evaluate,

since they are stated explicitly in the contract, usually in the form of a certain dollar quantity per unit time the job is tardy. It should be noted, however, that the existence of a penalty clause in a contract does not preclude the presence of the other costs listed if the job is, in fact, either tardy or early.

Costs of dealing with the customer are the costs incurred due to extra communication or deliberation with the customer when a job is delivered before or after its due date. Some of these may be direct, while others fall into the category of the opportunity cost of, for example, the manager using his time appeasing the customer when he could be allocating his time to other activities.

The cost of expediting tardy jobs might include the cost of hiring new men or administrative costs. If a job is completed early, it may be delivered early, in which case the supplier may incur extra costs of dealing with the customer and may risk customer dissatisfaction, or it may be delivered at the pre-established due date, in which case the supplier incurs an inventory holding cost. Exactly which policy is followed depends entirely on the operating conditions, but in the present research it is assumed that the product is held until the due date, and the cost for this has been included in the job waiting cost.

It is very difficult to place a monetary value on customer dissatisfaction or loss of good will. Possibly the customer is not offended by a tardy or early delivery. Alternately, the customer may cease to do business with the supplier because of a single tardy delivery. More often the consequences of a late delivery are somewhere between the above two extremes. Needless to say, because the losses can be severe,

it is certainly important for the manager to consider a penalty cost for late jobs in the scheduling decision.

The exact form of the aggregate of these lateness costs is not known. It is not unreasonable to assume that it could at least be estimated for a particular firm given its operating data. McNaughton [50] suggests the use of a quadratic loss function for tardy jobs since the urgency of getting a task done increases as time increases beyond the due date. Most authors have used a linear loss function when scheduling with due date considerations. Lawler [45] suggests the use of a penalty function which is monotonically nondecreasing with time. It would be unreasonable to assume that the penalty cost should decrease with increasing tardiness. It would also be unreasonable, under most circumstances, to assume that penalty cost continues to increase forever as tardiness approaches infinity.

Upon consideration of the above factors, the job penalty cost is formulated as follows. Let T_i' be the tardiness of job i, δ_i be the job penalty cost of job i, and a_{il}, $l = 1, 2, \ldots n$, be constants, then

$$\delta_i = a_{i1} T_i' + a_{i2} T_i'^2 + \ldots + a_{in} T_i'^n$$

$$= \sum_{l=1}^{n} a_{il} T_i'^l ; \quad i = 1, 2, \ldots, J \qquad (7)$$

Here n and the a_{il} are determined as desirable for the particular situation. If only a linear penalty cost is desired n = 1. In addition, an upper limit might be placed on δ_i so that the penalty remains reasonable for any value of tardiness. The job penalty cost component, $\delta(S)$, for schedule S is then:

$$\delta(S) = \sum_{i=1}^{J} \sum_{\ell=1}^{n} a_{i\ell} T_i^{\ell} \tag{8}$$

Total Opportunity Cost. An equation for total opportunity cost, TC, may now be formulated such that:

$$TC(S) = \beta(S) + \gamma(S) + \delta(S)$$

$$= \sum_{i=1}^{J} \sum_{n=1}^{g_i+1} rV_{in}y_{in} + \sum_{k=1}^{M} r_k[T(S) - \sum_{i,j} t_{ijk}]$$

$$+ \sum_{i=1}^{J} \sum_{\ell=1}^{n} a_{i\ell}T_i^{\ell} \tag{9}$$

Summarizing, the assumptions upon which the above function is based, in addition to those listed in Chapter I, include:

1. Processing costs as well as processing times are independent of sequence,

2. Job waiting costs are directly proportional to the value of the product at any time,

3. Jobs completed before their due date are held for delivery at the due date,

4. Machine idle costs are incurred for all machine inactive time from the start of the first job until the completion of the last job in the batch, and

5. The job penalty cost can be formulated as a polynomial function of tardiness. These assumptions are reasonable for most situations in which batch scheduling would be used. The developments which follow are based on the above assumptions. The algorithm eventually presented can be easily modified to accomodate the relaxation of the last three of the above five assumptions, however.

It should be mentioned here that one real cost to be considered has not been included in the total cost formulation and must be accounted for separately. This is the cost of implementing the scheduling analysis itself, which should, of course, be considered when comparing the methods available.

It is not claimed that it will always be possible to represent costs in the above manner. For example, the penalty costs may not be quantifiable at all, in which case the manager would consider the alternatives available to him and select the one which seemed best. However, since even a small scheduling problem involves thousands of possible alternatives, a suitable algorithmic approach seems desirable, if not

in fact, vital.

Characteristics of Economically Generated Schedules

A logical question concerning economically generated schedules arises with reference to characteristics of schedules which might be generated using economic indices. These characteristics can be analyzed in much the same manner as characteristics of time oriented measures are analyzed. It is well known, for example, that so called active schedules [30] dominate the set of all schedules when the criterion is schedule time.

Recall the following assumptions:

1. Machine idle time includes all machine inactive time from time zero until the last operation from the set of all jobs is completed,

2. Machine idle cost rate, r_k, for a given machine is constant over the scheduling period,

3. The penalty function for each job is a monotone non-decreasing function of completion time, and

4. Any operation adds value to the product on which it is performed.

Consider the following theorem:

Theorem 1: The set of active schedules contains a schedule, S, having minimum total machine idle cost and job penalty cost over all the feasible schedules.

Proof: Suppose S is a feasible nonactive schedule. Then by a series of left shift operations, an active schedule S' can be produced which is at least as good as schedule S, since the machine idle time will either decrease or remain the same, and the job completion times

will either decrease or remain the same. Hence if S has minimum possible total machine idle cost and job penalty cost, it is from the set of active schedules.

This suggests that when it is desired to produce schedules which are good with respect to total machine idle cost and job penalty cost, one good procedure would be to search from among the set of active schedules in much the same manner as active schedules are searched in order to find good schedule time solutions. However examination of the remaining cost component, job waiting cost, shows that it does not behave in the same manner. Left shifting of an operation through a time interval actually increases the job waiting cost since the time during which the job has its higher value is increased while the time during which the job has its lower value is decreased. Consider the following theorem:

Theorem 2: Given a schedule S such that some operation can be right shifted without the left shift of another operation in order that a schedule S' be obtained such that:

$$T_i'(S') \leq T_i'(S) \; ; \; i = 1, 2, \ldots, J \qquad (10)$$

$$T(S') \leq T(S) \qquad (11)$$

then $TC(S') < TC(S)$.

Equation (10) prevents delay of the last operation of a job when this would increase tardiness. Equation (11) prevents delay of operations

past the time which would increase schedule time. Proof is given only for the case where resequencing does not occur. A similar but more involved proof could be given for the case where an operation is delayed past another operation yielding a different sequence.

Proof: Since the conditions of equations (10) and (11) prevent an increase in machine idle costs and job penalty costs, it remains only to be shown that the procedure decreases job waiting cost. Consider job a.

$$\beta_a = r \sum_{n=1}^{g_a+1} V_{an}\, y_{an} \tag{12}$$

Let operation ℓ of job a be right shifted, $\ell \in \{1, 2, \ldots, \ell - 1, \ell + 2, \ldots, g_a\}$. Then

$$y'_{a\ell} > y_{a\ell} \quad \text{and} \quad y'_{a(\ell+1)} < y_{a(\ell+1)}$$

and $y_{aj} = y'_{aj}$ for $j \in \{1, 2, \ldots, \ell - 1, \ell + 2, \ldots, g_a\}$. Also $V_{aj} = V'_{aj}; \forall_j$. However,

$$y'_{a\ell} + y'_{a(\ell+1)} = y_{a\ell} + y_{a(\ell+1)} \tag{13}$$

$$y_{a(\ell+1)} - y'_{a(\ell+1)} = y'_{a\ell} - y_{a\ell} > 0 \tag{14}$$

Then since $V_{a(\ell+1)} > V_{a\ell}$,

$$V_{a(\ell+1)} \left[y_{a(\ell+1)} - y'_{a(\ell+1)}\right] > V_{a\ell} \left[y'_{a\ell} - y_{a\ell}\right] \tag{15}$$

$$V_{a\ell}y'_{a\ell} + V_{a(\ell+1)}y'_{a(\ell+1)} < V_{a\ell}y_{a\ell} + V_{a(\ell+1)}y_{a(\ell+1)} \tag{16}$$

$$\rightarrow \beta'_a < \beta_a$$

It is seen that shifting operations either left or right has an effect on job waiting cost antithetic to the effect on machine idle and job penalty cost. The conditions for a minimal total cost solution are not immediately apparent, nor is a procedure for obtaining a minimal total cost solution. Therefore, a solution procedure is proposed whereby a schedule having minimal total machine idle cost and job penalty cost is obtained by searching from the set of active schedules, followed by a rescheduling of operations in the manner suggested by Theorem 2 to reduce job waiting cost. Although it can easily be shown that this procedure does not necessarily produce an optimal total cost solution, it is quite likely to produce a solution which is very close to optimal, especially if the problem parameters are such that job waiting cost is

a small portion of the total cost.

Development of the Algorithm

Potential approaches for solution of static scheduling and sequencing problems include enumeration, sampling, and algorithmic techniques. Enumeration, which will produce the optimal solution, is impractical even in the time domain for other than very small problems. By the same token, sampling, which does not guarantee optimality, has not received wide acceptance. Neither would be practical when considering the economic factors because active schedules do not dominate. The number of possible schedules is unlimited since any number of schedules can exist for a given sequence. The algorithmic structure presented is a branch and bound approach similar to that first proposed by Brooks and White [11]. A graph theoretic approach similar to that in references [6] and [27] facilitates the process of branching, while a lower bound on machine idle cost and job penalty cost provides the bounds by which solution sets are eliminated. The graph theoretic approach is chosen because recent studies, such as [27], indicate that it may offer a computational advantage over earlier techniques. It should be emphasized that the procedure is heuristic and cannot, even with backtracking, guarantee an optimal solution.

The Graph Structure of the Shop Scheduling Problem

The static shop scheduling problem can be structured as a mixed graph,* $G(\mathcal{N}, \mathcal{C}, \mathcal{D})$ [6, 13] with the nodes of the graph representing

* Graph theory terms used in this work are defined in the glossary given in Appendix B.

operations and the arcs representing precedence relationships. This configuration is convenient since arrangement of arcs can be used to indicate a sequence, giving precedence relationships among nodes. Conjunctive arcs, those in the set $\mathcal{C}$, represent machine ordering restrictions while disjunctive arc pairs, given by $\mathcal{D}$, represent the potential ordering of two operations on a machine. The set of all arcs is given by $\mathcal{A}$, so that $\mathcal{C} \cup \mathcal{D} = \mathcal{A}$, and $\mathcal{C} \cap \mathcal{D} = \phi$, the empty set. Observe Figure 1 showing the initial graph with conjunctive arcs as solid lines and disjunctive arcs as dashed lines. Each time one operation is selected to precede

node	1	2	3	4	5	6	7	8	9
job	1	1	1	2	2	2	3	3	3
machine	3	1	2	1	3	2	3	2	1

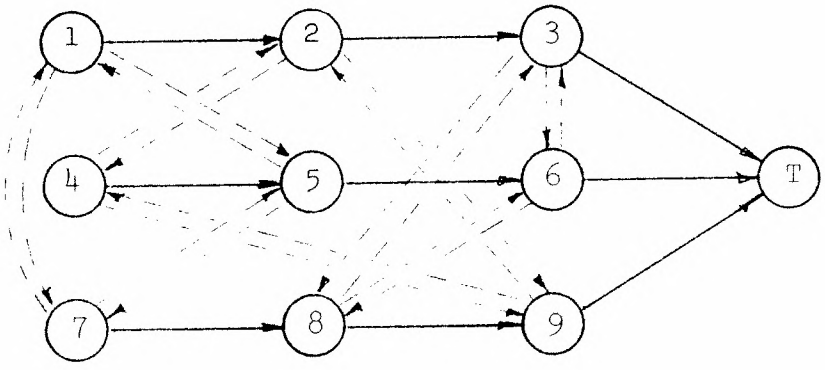


Figure 1. 3 X 3 Problem Showing Conjunctive and Disjunctive Arcs

another, one arc is selected from a disjunctive pair to indicate the appropriate precedence relationship. After proceeding iteratively, eventually (J - 1) arcs are selected for each machine so that M arborescences result which represent the job sequencing on the machines, as in Figure 2. For the case where each job is processed once and only once on each machine, it is seen that initially $|\mathcal{C}| = JM$, and $|\mathcal{D}| = J \cdot \binom{M}{2}$.

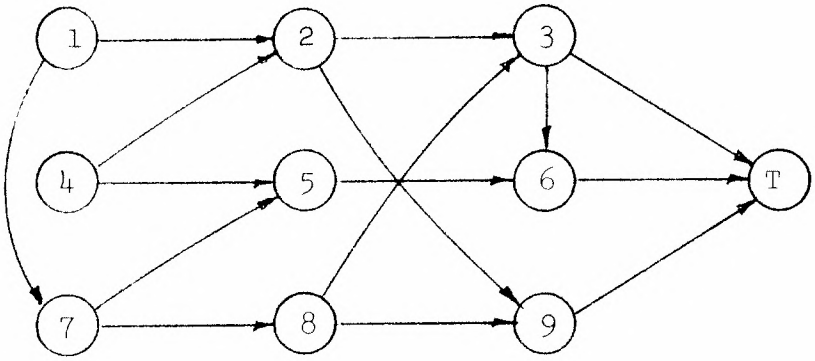


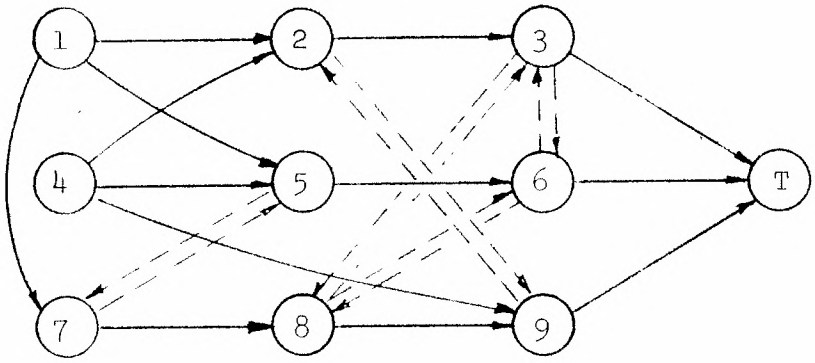
Figure 2. A Feasible Sequence for the 3 X 3 Problem

In general, a synthesis of the initial structure, $G_o(\mathcal{N}, \mathcal{C}, \mathcal{D})$, is to be obtained such that some $G_p(\mathcal{N}, \mathcal{C})$ results where $|\mathcal{D}| \rightarrow \phi$ after some number of iterations, p.

Identification of Candidate and Conflict Sets

When an operation is scheduled all disjunctive arcs incident to

that node are elinimated. The start time and completion time of that operation in the eventual schedule are then known, assuming the earliest possible start time is assigned. An operation is schedulable, or is a candidate for scheduling, if it has not been scheduled but all operations constrained by machine ordering restrictions to precede it have been scheduled. At the beginning of each iteration a candidate set, Ψ, consisting of all the schedulable operations, is identified, as in Figure 3. A member of a candidate set which is the only member processed on



Scheduled operations = {1, 4}

Ψ = {2, 5, 7}

θ = {5, 7}

Figure 3. 3 X 3 Problem at Some Iteration

a particular machine will be referred to as a singleton, and it may be scheduled immediately. If, however, two or more members are to be processed on the same machine, then a conflict set, θ, exists and one of the operations must be selected to precede the others. The operation so selected is then scheduled and the remaining operation or operations become a member of the next candidate set. A candidate set may have more than one conflict set, and a conflict set can have up to J operations.

Selection of the operation of the conflict set to be scheduled has usually been done by establishing a lower bound on some measure of performance, such as schedule time, contingent upon that particular operation being scheduled. The operation from the conflict set exhibiting the least lower bound is selected.

The Cost Bound

A resolution of conflicts based on a lower bound on the sum of machine idle cost and job penalty cost is proposed. The resulting schedule will then have characteristics as described previously. Consider the following definitions:

$(ijk) \equiv$ operation j of job i on machine k

$B^M_{ijk} \equiv$ lower bound on total machine idle cost incurred if (ijk) is schedule on machine k before other operations in conflict set.

$B^P_{ijk} \equiv$ lower bound on total job penalty cost incurred if (ijk) is scheduled on machine k before other operations in conflict set.

$B_{ijk} \equiv B^M_{ijk} + B^P_{ijk}$

$\theta \equiv$ a conflict set

$\eta \equiv$ set of unscheduled operations

s_{ijk} ≡ earliest start time of (ijk)

C_{ijk} ≡ completion time of (ijk)

$t_{\cdot\cdot k}$ ≡ sum of processing times of all operations performed on machine k

MB_{ijk} ≡ machine based lower bound on schedule time if (ijk) is scheduled on machine k before other operations in conflict set

d_i ≡ due date of job i.

The machine based bound may be calculated as:

$$MB_{mn\ell} = \max \left\{ \left[C_{mn\ell} + \sum_{\substack{ij\ell \epsilon \eta \\ ij \neq mn}} t_{ij\ell} \right], \left(\max_{\substack{k' \\ k' \neq \ell}} \left[\min_{ijk' \epsilon \eta} (s_{ijk'}) \right. \right. \right. \quad (17)$$

$$\left. \left. \left. + \sum_{ijk' \epsilon \eta} t_{ijk'} \right] \right) \right\}$$

where the terms in the first set of brackets represent the minimum schedule time due to interference on the machine of conflict and the terms in the second set of brackets represent the minimum schedule time due to interference on other machines. A lower bound on tardiness of job i can be calculated as:

$$T'_i = \max [0, C_i - d_i] \; ; \; i = 1, 2, \ldots, J \qquad (18)$$

where C_i results as a lower bound on completion time for job i. Now the lower bounds on cost may be given as:

$$B^M_{mn\ell} = \sum_{k=1}^{M} r_k \, (MB_{mn\ell} - t..._k) \tag{19}$$

$$B^P_{mn\ell} = \sum_{i=1}^{J} (a_{i1} \, T_i' + a_{i2} \, T_i'^2) \tag{20}$$

$$B_{mn\ell} = B^M_{mn\ell} + B^P_{mn\ell} \tag{21}$$

Here only the linear and quadratic terms for penalty cost have been shown. Higher order terms could, of course, be included as shown previously, as could an upper limit on penalty cost. Note that the cost bounds are formulated from bounds on time measures.

In review, the algorithm involves an iterative process. At each iteration a candidate set is identified. From this candidate set the singletons are scheduled and conflict sets are identified. Within each conflict set, lower bounds are calculated based on selection of each operation by temporarily changing the graph to reflect selection of that operation for scheduling, and the operation yielding the least lower bound is selected. At the end of each iteration, the selected operations from the conflict sets are scheduled by selection of the appropriate disjunctive arcs, leaving an updated graph for the next iteration. When the last operation has been scheduled, the process is complete and the total cost may be calculated.

Additional Features

The procedure described above may be used in a first pass solution mode or in a backtracking mode. The first pass solution is available immediately when the last operations have been scheduled. To obtain a solution which may be better with respect to total machine idle cost and job penalty cost, it is necessary to backtrack up the solution tree of conflicts until a lower bound is formed which is less than the total machine idle and job penalty cost obtained on the first pass solution, as in Figure 8. This conflict resolution is then pursued until it produces a lower bound greater than the current best solution or a new lower cost solution. All points on the solution tree are thus explored. It is noted that this backtracking will not necessarily produce a solution which is optimal with respect to total machine idle and job penalty cost because some feasible active schedules are not represented in the tree. This is due to the fact that the time element is not considered when the candidate sets are designated. An indication of how this algorithm might be altered so that the optimal total machine idle and job penalty cost could be guaranteed by backtracking is found in reference [4].

A separate feature of the algorithm is a delay scheme which delays operations to reduce job waiting costs. The solution obtained thus far may have idle time after operations which allows the delay of the operations without violating ordering restrictions. Such operations are delayed as far as possible subject to no resequencing, no increase in machine idle cost, and no increase in job penalty cost, thus giving a lower job waiting cost.

An additional feature is incorporated in the algorithmic structure which holds singletons from scheduling at a given iteration if an operation in the next possible candidate set is processed on the same machine. This offers a possibility of an improved first pass solution [6].

Computational Algorithm

Following is a step by step procedure for the first pass cost algorithm embodying the features described in the preceding section. Additional notation which will be used is introduced as

- (k) - node k
- (k, ℓ) - arc directed from (k) to (ℓ)
- $\mathcal{N}$ - set of all nodes
- $\mathcal{M}$ - machine ordering matrix
- $\mathcal{M}_j$ - vector of operations in job j; row j of $\mathcal{M}$
- $\mathcal{J}_m$ - vector of operations processed on machine m
- $\mathcal{N}_1$ - set of scheduled operations
- $\mathcal{N}_2$ - set of unscheduled operations
- σ - set of singleton operations

STEP 1: Initialize the precedence graph

1.1 Construct J linear graphs, from the machine orderings such that

$$(k) \ll (\ell) \ll \ldots \ll (T) \mid (k), (\ell) \in \mathcal{M}_j; \forall_j, (k,\ell) \in \mathcal{E}$$

where (T) is the terminal node.

1.2 Label all arcs in $\mathcal{A}$ such that

$f(k, \ell)$ = processing time, operation k; (k) = 1, 2, . . . N

1.3 Select the initial set of arcs (k, ℓ) such that

$$(k, \ell) \in \mathcal{D};\ (k),\ (\ell) \in \mathcal{J}_m;\ \forall_m$$

1.4 Set p = 0, $\Psi_p = \{\phi\}$, $\mathcal{N}_1 = \phi$, $\mathcal{N}_2 = \mathcal{N}$.

STEP 2: Construct the candidate set Ψ_p.

2.1 Set p = p + 1 and scan each linear graph for nodes (k) such that

$$(\ell) \ll (k),\ (\ell) \in \mathcal{N}_1,\ (k) \in \mathcal{N}_2$$

Set (k) into Ψ_p. If p = 1, go to 2.3.

2.2 Update Ψ_p such that nodes (χ) are removed, where

$$(k) \in \Psi_p;\ (k) \ll (h);\ (h),\ (\chi) \in \mathcal{J}_m, \mathcal{N}_2,$$

and (χ) is the only node from specific $\mathcal{J}_m$ in Ψ_p.

2.3 Identify singletons. Set all nodes (χ), where (χ) is the only node from specific $\mathcal{J}_m$ in Ψ_p, into the set σ, and remove from Ψ_p.

2.4 Schedule nodes in σ. Update $\mathcal{N}_1$ and $\mathcal{N}_2$.

2.5 Update the precedence graph. Let (q) be the scheduled node, then:

2.5.1 If $G_p(\mathcal{N},\mathcal{A})$ is unsaturated over given $\mathcal{J}_m$, update the graph such that

$$\mathcal{A} := \mathcal{A} + (q, \ell),\ (q) \in \mathcal{J}_m,\ (\ell) \in \mathcal{J}_m, \mathcal{N}_2$$

2.5.2 If $G_p(\mathcal{N},\mathcal{A})$ is saturated over given $\mathcal{J}_m$, update the graph such that

$$\mathcal{A} := \mathcal{A} + (q, \ell),\ (q) \in \mathcal{J}_m,\ (\ell) \in \mathcal{J}_m, \mathcal{N}_2$$

and

$$\mathcal{A} := \mathcal{A} - (h, \chi),\ (h) \in \mathcal{J}_m,\ (\chi) \in \mathcal{J}_m, \mathcal{N}_2,\ (h) \neq (q)$$

2.6 If p = 1, go to 2.1.

STEP 3: Identify conflicts and select nodes for scheduling

3.1 Identify a conflict set. If more than one node (k) from a given $\mathcal{J}_m$ exists in Ψ_p, resolve the conflict by Step 3.2.

3.2 Arrange the graph for the scheduling of a conflict set node, as in step 2.5. Calculate the lower bound on total machine idle and job penalty cost in accordance with equations (17) through (21). Return the graph to its former state. Repeat for each node in the conflict set.

3.3 Place the node with the least lower bound into the set of operations to be scheduled at this iteration. In case of a tie the lowest numbered node is selected.

3.4 If the last conflict set in the candidate set Ψ_p has been resolved, go to step 4; otherwise return to step 3.1.

STEP 4: Schedule the selected operations and update the graph.

4.1 Place the nodes selected in step 3 into $\mathcal{N}_1$; set $\mathcal{N}_2 = \mathcal{N} - \mathcal{N}_1$. Update graph as in 2.5.

4.2 If all operations have been scheduled, $\mathcal{N}_2 = \phi$, go to step 5; otherwise return to step 2.

STEP 5: Evaluate the total cost of the schedule.

STEP 6: Delay operations to reduce job waiting costs.

STEP 7: Evaluate the new waiting cost and total cost.

Since at least one operation is selected for scheduling at each iteration, the algorithm will converge after a finite number of iterations. The algorithm is now demonstrated with a sample problem.

Sample Problem

Consider the following 4 X 3 problem. The machine ordering and operation processing times can be given by the matrices $\mathcal{M}$ and τ such that

$$\mathcal{M} = \begin{vmatrix} 12 & 13 & 11 \\ 21 & 23 & 22 \\ 33 & 31 & 32 \\ 41 & 42 & 43 \end{vmatrix} \qquad \tau = \begin{vmatrix} 4 & 2 & 3 \\ 8 & 4 & 5 \\ 6 & 3 & 9 \\ 7 & 6 & 2 \end{vmatrix}$$

The due date matrix is given as:

$$[d_i] = \begin{vmatrix} 9 \\ 37 \\ 43 \\ 42 \end{vmatrix}$$

The machine idle cost rates are given by:

$$[r_k] = [30,\ 70,\ 90]$$

The job penalty cost rates are given by:

$$[a_{i1}] = \begin{vmatrix} 90 \\ 80 \\ 80 \\ 20 \end{vmatrix} \quad [a_{i2}] = \begin{vmatrix} 9 \\ 8 \\ 8 \\ 2 \end{vmatrix}$$

The job value matrix is:

$$V = \begin{vmatrix} 1000 & 1200 & 1300 & 1450 \\ 300 & 940 & 1300 & 1600 \\ 200 & 740 & 1010 & 1820 \\ 300 & 790 & 1090 & 1230 \end{vmatrix}$$

where the figures in the job value matrix give total value rather than the value added in the respective operations. The waiting cost rate,

r, is given as .00041 dollars per dollar value held for one day. All other units are the appropriate combination of dollars and days. The subscript j is omitted in the example since each job is processed only once on each machine. Given the above information the algorithm proceeds as follows:

STEP 1: Initialize the precedence graph. The precedence graph can be constructed as shown in Figure 4, where there is a one to one correspondence between nodes (k) and operations in $\mathcal{M}$.

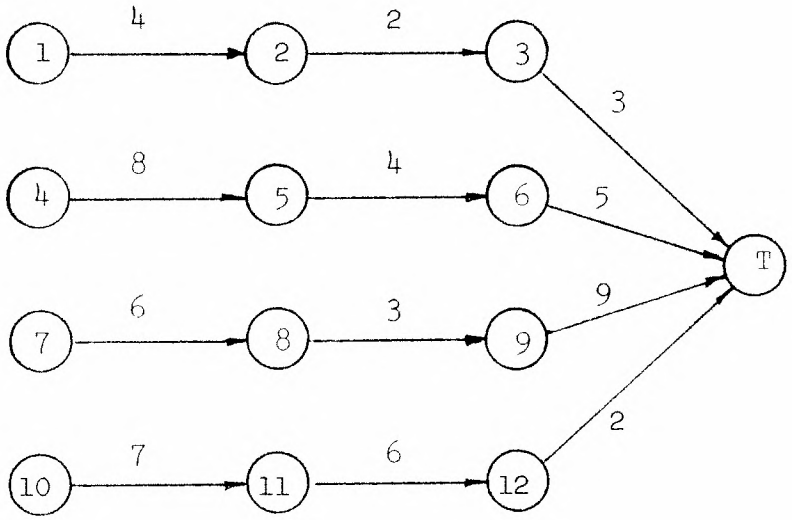


Figure 4. Sample Problem Initial Precedence Graph

Note that in Figure 4, only conjunctive arcs are included initially. The iteration index, p, is set at zero.

STEP 2: Construct the candidate set Ψ_p. The iteration index is incremented to 1. The candidates are identified as nodes (1), (4), (7), and (10), of which (1) and (7) are singletons. Since this is the first

iteration both operations are scheduled and the graph is updated as shown in Figure 5.

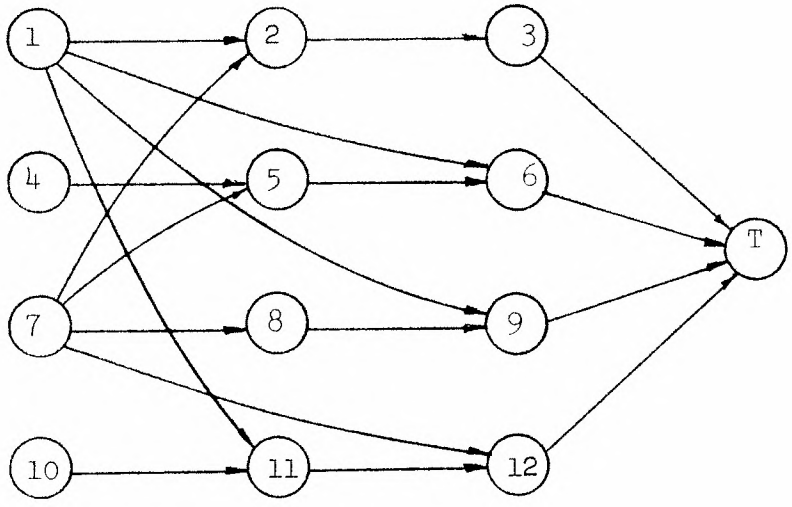


Figure 5. Graph at First Iteration

At this point the iteration index p is set at 2 and the new set of candidates identified as nodes (2), (4), (8), and (10). Node (2) is a singleton and is held over since node (5), which like node (2) is processed on machine 3, can appear in the next candidate set. Hence the final candidate set at the current iteration is nodes (4), (8), and (10).

STEP 3: Identify conflicts and select nodes for scheduling. Since operations (4), (8), and (10) are all processed on machine 1, there exists only one conflict set in the set of candidate operations. The arcs of the graph are arranged to correspond to the selection of node (4), the first of the three operations, as shown in Figure 6. Noting

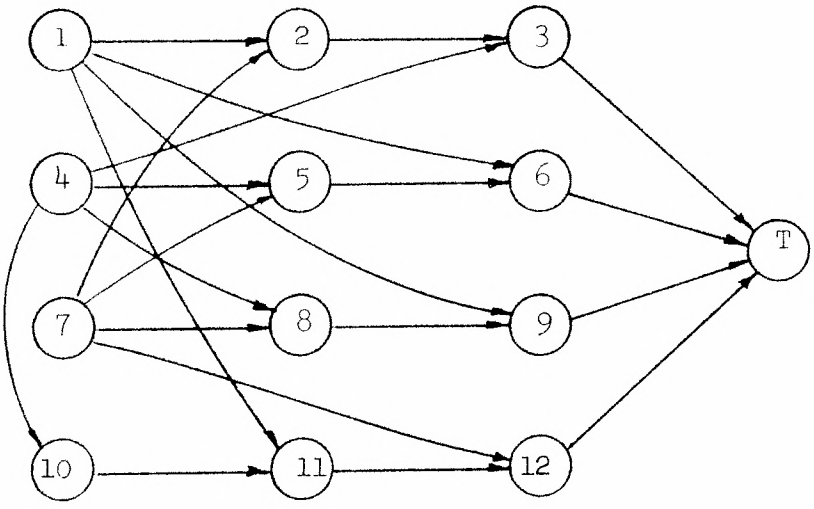


Figure 6. Graph at Second Iteration, Given the Selection of Operation (4)

that node (4) corresponds to job 2 on machine 1, the lower bound B_{21} may now be calculated. The reader can verify the following values as calculated from formulations (17) through (21) given in a preceding section.

$$MB_{21} = \max[8 + 13, \min(12, 11, 15) + 20, \min(6, 8, 21) + 8] = 31$$

$$T_1' = \max[0, 11 - 9] = 2$$

$$T_2' = \max[0, 17 - 37] = 0$$

$$T_3' = \max[0, 20 - 43] = 0$$

$$T_4' = \max[0, 23 - 42] = 0$$

$$B_{21}^M = 30(31 - 21) + 70(31 - 24) + 90(31 - 14) = 2320$$

$$B_{21}^P = 90(2) + 9(2)^2 = 216$$

$$B_{21} = 2536$$

The arcs of the graph are now arranged to correspond to the scheduling of operation (8), the next operation of the conflict set, whereupon it is found that $B_{31} = 2291$. Similarly, it is found that $B_{41} = 1776$. Since all lower bounds for the conflict set have been calculated, operation (10), the one having the least lower bound, is placed into the set of operations to be scheduled. Since there was only one conflict set in the candidate set, the computation proceeds to step 4.

STEP 4: Schedule the selected operations and update the graph. Operation (10) is scheduled. The updated graph appears as in Figure 7.

At this point the algorithm returns to step 2, commencing with the next iteration. This procedure continues in a similar manner through nine iterations, after which time the first pass solution is reached with a total machine idle and job penalty cost of $2,634. The entire computation is summarized and presented in Table 1.

STEP 5: Evaluate the total cost of the schedule. When the costs are evaluated in accordance with equations (3), (6), (8), and (9), it is

Table 1. Complete Solution to Sample Problem

Iter.	Cand. Set	Single-tons	Held Over	Conflict Sets	Lower Bounds	Least Lower Bounds
1	1,4,7,10	1,7	-	-	-	-
2	2,4,8,10	2	2	4	2536	
				8	2291	
				10	1776*	1776
3	2,4,8,11	2,11	2,11	4	3099	
				8	2064*	2064
4	2,4,9,11	2,4	2,4	9	2634	
				11	2064*	2064
5	2,4,9,12	4,9	4	2	2064*	2064
				12	3639	
6	3,4,12	12	12	3	2634*	2634
				4	3936	
7	4,12	4,12	12	-	-	-
8	5,12	-	-	5	2634*	2634
				12	2634	
9	6,12	6,12	-	-	-	-
			Backtracking			
10	2,4,8,10	2	2	4	2536	
				8	2291*	
				10	1776+	1776
11	2,4,9,10	2,9	2,9	4	4019x	
				10	3740x	3740x
12	2,5,8,10	-	-	8	2995x	2995x
				10	4049x	Solution Terminates

* Node selected for branching
\+ For backtracking, implies node already explored
x Lower bound higher than best solution, branch terminates

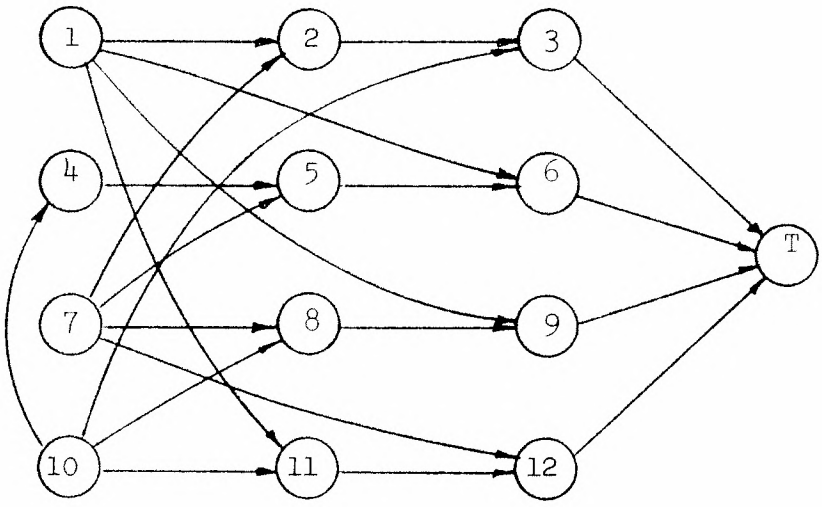


Figure 7. Graph at End of Iteration 2

found that job waiting cost is $38.38, machine idle cost is $2,130.00, job penalty cost is $504.00, and total cost is $2,672.38.

STEP 6: Delay operations to reduce job waiting costs. The following operations are delayed:

Job 1 on machine 2 by 4 time units

Job 1 on machine 3 by 2 time units

Job 3 on machine 3 by 1 time units

Job 3 on machine 2 by 3 time units

Job 4 on machine 2 by 3 time units

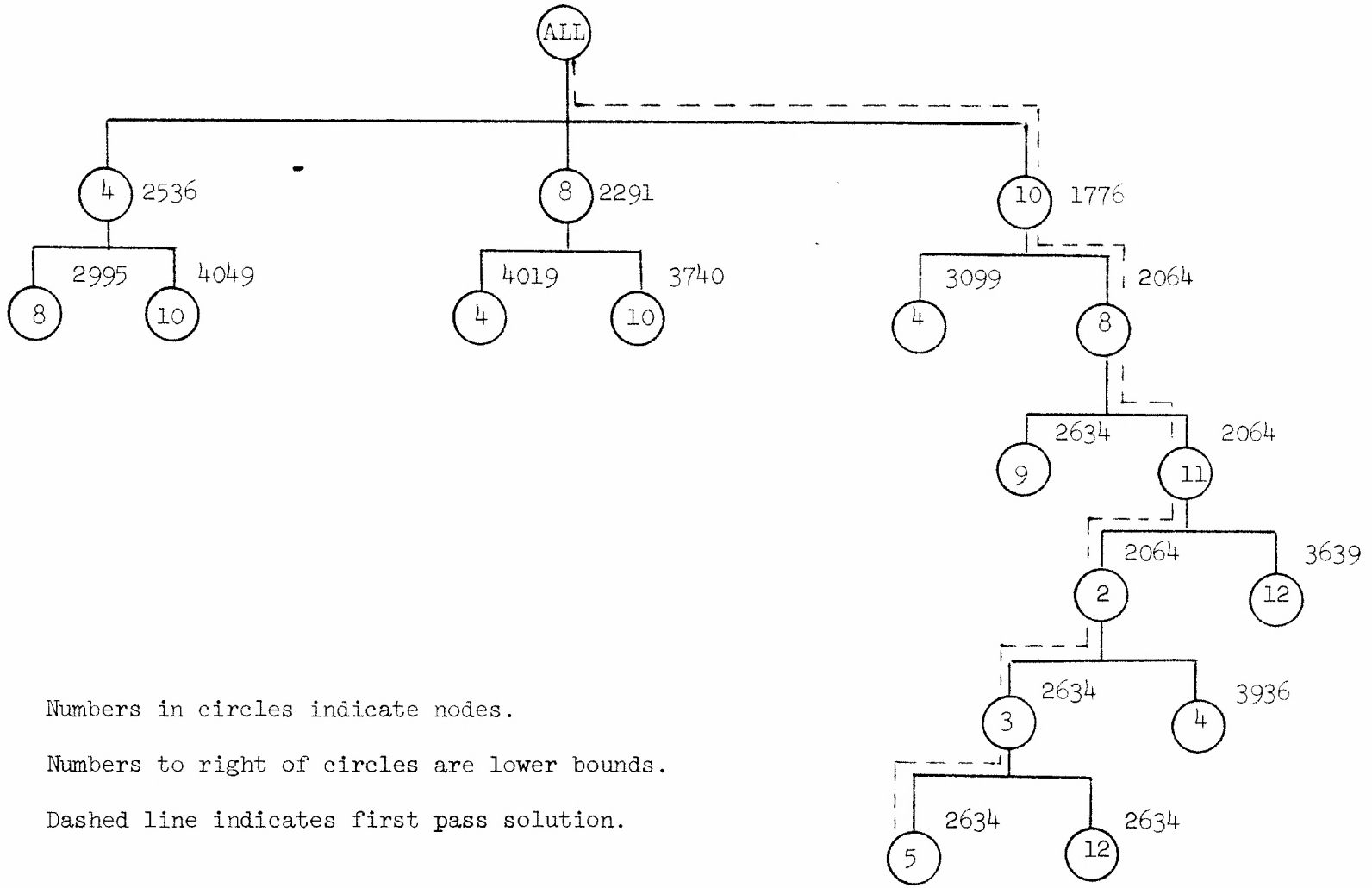
Job 4 on machine 3 by 3 time units

STEP 7: Evaluate the new waiting cost and total cost. The new job waiting cost is found to be $36.22 and the total cost is $2,670.22.

This completes the first pass solution procedure. It is

appropriate at this point to mention that, as explained earlier, backtracking may be used to improve upon the first pass solution. Even though the computational algorithm as stated above does not provide for backtracking, the results of using a backtracking solution for the sample problem are shown in Table 1. In this problem backtracking does not yield a better solution. As explained in the section on development of the algorithm, backtracking does not guarantee optimality. Figure 8 shows the solution tree for the sample problem including backtracking.

The algorithm presented above was made the subject of extensive experimental investigation. The nature of the investigation as well as its resultant is presented in the succeeding chapter.



Numbers in circles indicate nodes.

Numbers to right of circles are lower bounds.

Dashed line indicates first pass solution.

Figure 8. Solution Tree for Example Problem

CHAPTER III

EXPERIMENTAL INVESTIGATION OF THE COST ALGORITHM

In order to evaluate the merit of the cost algorithm computer experimentation was undertaken. The prime objectives of the experimentation were to investigate the worth of the cost algorithm in comparison with some existing scheduling procedures and to test the sensitivity of the algorithm with reference to selected parametric variations. The algorithm was coded in FORTRAN V language for use on the UNIVAC 1108 computer at the Georgia Institute of Technology. A source listing is provided in Appendix C.

Procedure and Experimental Design

The cost bounding algorithm was tested in the first pass mode only. Since no algorithms on cost solutions are available from the literature and moreover, since some comparison with existing procedures was desired, a heuristic schedule time algorithm was selected initially as a standard of measure. The particular algorithm chosen was a heuristic branch and bound using a composite of machine based and job based lower bounds on schedule time [4]. The algorithm is given for reference in Appendix A. This algorithm is structured in much the same manner as the cost algorithm, the major difference being in the nature of the lower bound itself.

Initially a bounding procedure was developed which was, for both the job-machine based algorithm and cost bound algorithm, somewhat less

powerful than the algorithms described in Chapter II and Appendix A. These original algorithms are hereafter referred to as the initial cost bound algorithm and the initial job-machine bound algorithm. The revised algorithms as presented earlier will be referred to as the cost bound algorithm and the job-machine bound algorithm. The initial job-machine bound employed the job based bound on schedule time only on the basis of the jobs in the conflict set of operations, while the subsequent job-machine bound employed the more powerful procedure of using the job based bound over the maximum of the earliest completion times of all jobs, as in Appendix A. Similarly the initial cost bound used a bound on tardiness of jobs in the conflict set, while the subsequent procedure used a bound on tardiness of all jobs as in Chapter II.

Further comparative evaluation was carried out using a slack per operation dispatching rule. This allowed the consideration of due dates in a scheduling procedure based on local rules without explicit calculation of penalty costs. The slack per operation rule was selected because, at least in the case of the dynamic problem, experiments have shown it to be somewhat better than other standard due date type priority rules [15].

Selection of Problems and Parameters

The problems used in the testing were selected from previous works, most of which are documented. The moderate size problems appeared in references [4] and [5]. The remaining problems came from various other published works listed in the Bibliography.

Since the problems available from the literature consider only the machine ordering and processing times, such parameters as machine

idle cost rate and due dates must be set separately. Certainly, some methods available for selecting these parameters would include attainment of values from accounting data in an actual job shop, arbitrarily selecting values which seem reasonable, and using a random process to generate values. Selection of values from an actual job shop would result in quite realistic values, but would involve extensive investigation for each problem, and might lead to results biased toward one particular kind of shop or operating environment. Arbitrary selection of values might also suffer from the threat of biased results. Hence a random process was used to generate parameter values over a range which might be encountered in a job shop. In particular, the parameters were set in the following manner:

1. Time units: 1 time unit → 1 day.

2. Due date, d_i: randomly selected from a uniform distribution on one to three times the total processing time of job i.

3. Job waiting cost rate, r: randomly selected from the values .15, .25, or .35 dollars per dollar-year held.

4. Raw material value, V_{i1}: randomly selected with uniform probability from the values \$100, 200, . . . , 1000.

5. Value added, v_{ij}: randomly selected with uniform probability from \$50, 60, 70, 80, or 90 times the processing time for the operation. Hence, value added is correlated to processing time.

6. Machine idle cost rate: r_k - randomly selected with uniform probability from the values 0, 10, . . . , 90 \$/day.

7. Job penalty cost rates: linear term, a_{i1} randomly selected from 0, 10, . . . , 90 \$/day. Quadratic term a_{i2} set at one tenth the

linear rate, $\$/day^2$. All other terms were set at zero.

Experimentation

The first experiment conducted, herein referred to as experiment A, was a direct comparison between the initial cost bound algorithm and initial job-machine bound algorithm. Nineteen problems, each with separately selected parameter sets were solved once by each of the two bounding procedures and the total cost of the resulting schedule calculated. These problems and parameter sets remained exactly the same for all experiments except as described in experiments D, E, and G. After the algorithms were updated to embody the form described in Chapter II as well as in Appendix A, the same set of problems were solved again as experiment B. The objective of this experimentation was to determine whether the most economical schedules would be produced using a solution procedure based on minimization of schedule time or using a procedure based on cost. For the seven smallest problems, the cost of an available schedule having the best known schedule time was also calculated for comparison.

Since the job-machine bound uses no due date information, further comparison of the cost bound with a procedure which does consider due date information was desirable. Hence, experiment C was generation of schedules using a slack per operation dispatching rule for the same nineteen problems. This algorithm operated in a similar manner to those above except that rather than resolving conflicts using a bound on cost or schedule time the conflicts were resolved in favor of the operation whose job had the least slack per operation remaining. Further, in order to include cost information in the decision rule, a weighted

slack per operation dispatching rule, which gives higher priority to a job having a higher penalty cost rate, was developed and tested.

In order to provide some experience with larger problems experiment D, consisting of one 20 X 10 and one 20 X 5 problem, was run using both the cost bound and the job-machine bound. The objective was not only to test algorithmic performance but also to get some idea of the computational effort required for larger problems using the cost bound.

There is no standard procedure available for testing the sensitivity of the cost algorithm to the cost parameters, and so simple tests were devised to give some insight into this sensitivity. One feature of interest is whether the differences between the solutions obtained using the cost bound and job-machine bound is primarily dependent on the problem or the parameters. To break out this effect one problem was selected and five sets of parameters were chosen. The problem was then solved with both bounds using the five new sets of parameters, providing a total of six replications including the original parameter set. This is referred to as experiment E.

Experiment F was designed to determine the effects of limiting the job penalty cost for a job to some predetermined value. As discussed in a preceding section, it is unreasonable to expect penalty cost to increase ad infinitum as tardiness increases. For this experiment, the maximum value of penalty cost was limited to twice the value of the completed job. For the cost bound this should change the schedule generated and thus the total cost, since limiting the penalty cost can cause conflicts to be resolved in a different way. In the case of the job-machine bound, however, the schedule should not change

since penalty cost does not enter into schedule determination. The costs generated for the job-machine bound should be different from those obtained in experiment B only when the penalty cost for at least one job attains the limiting value.

Experiment G was designed to determine sensitivity of the results to a change in due date setting policy. With all other parameters remaining the same, the due dates were changed to values obtained from a uniform distribution of two to five times the total processing time of the job instead of the one to three times the total processing time which was used for the other experiments.

Results

Results of experiments A through G are shown in Tables 2 through 9, where for each problem in each experiment the schedule time, total cost, and cost after delay are indicated. The total cost referred to is the sum of the job waiting cost, machine idle cost, and job penalty cost of the active schedule resulting from the algorithmic first pass solution. The cost after delay is the total cost resulting when operations are delayed to reduce waiting cost as described in Chapter II. For experiment B only, the cost components are broken out individually so that the reader may gain appreciation for the relative sizes of the costs.

Since the magnitudes of the resulting total costs have no meaning when comparing one problem to another, a nonparametric sign test is used to establish statistical significance in each case where direct comparison between procedures is desired. Even though delay of operations to reduce job waiting cost would not ordinarily be used when

schedules are generated by a time based algorithm like the job-machine bound, statistical comparison is between cost after delay for both algorithms to maintain conservatism.

Experiment A

For the nineteen problems tested using the initial algorithms, as shown in Table 2, the cost bound gave the lowest cost schedule in 14 problems, the job-machine bound algorithm gave the lowest cost schedule in three, and identical schedules were obtained for two. Hence, for a sign test [54], $n = 17$, $z = 3$, and the hypothesis that the algorithms are equally efficient in producing good economic schedules, is rejected at the $\alpha = .05$ level, with $.01 < \hat{\alpha} < .05$.

Experiment B

The results arising from the analysis of the same nineteen test problems using the algorithms of Chapter II and Appendix A are shown in Table 3. The cost bound gave the best solution in thirteen problems, the job-machine bound was best for four, and there were two ties. Hence $n = 17$, $z = 4$, and the cost algorithm is judged superior at the $\alpha = .05$ level with $\hat{\alpha} = .05$.

If Tables 2 and 3 are compared it can be seen that for the job-machine bound the revised algorithm gives one schedule superior and one schedule inferior to the initial algorithm, and for the cost bound the revised algorithm produced ten schedules superior and two schedules inferior to the initial algorithm. Note also that job penalty cost makes up, by far, the largest portion of the cost for most problems.

Table 4 shows the total cost with no delay of operations for a solution known to be optimal with respect to schedule time, for the

seven smallest problems. Comparison of Tables 3 and 4 shows that, when delay is not considered, the cost bound produces a better schedule for two problems and the same schedule for one problem.

Experiment C

The slack per operation dispatching rule produced one solution better than the cost algorithm and one identical schedule. Hence $n = 18$, $z = 1$, and the cost bound produces better solutions at the $\alpha = .05$ level, with $\hat{\alpha} < .01$. As Table 5 shows, the weighted slack per operation rule consistently performs better than the non-weighted rule. However, when compared with the cost bound the weighted rule gives the same statistical results as the non-weighted rule.

Experiment D

The cost bound gave lower cost solutions than the job-machine bound for both of the two larger problems. Computer run times for the two algorithms were comparable. The 20 X 10 problem required 195.7 seconds for the job-machine bound and 215.7 seconds for the cost bound, while the 20 X 5 problem required 48.3 seconds for the job-machine bound and 40.9 seconds for the cost bound.

Experiment E

It was observed that a change of parameters did in fact cause a change in results. Of the five new sets of parameters, the job-machine bound gave a better solution in two cases and the cost bound gave a better solution in three cases.

Experiment F

Establishing an upper limit on job penalty cost caused the cost bound to produce different schedules from those of experiment B in

seven of the nineteen problems. Table 8 shows that the cost bound gave a better schedule than the job-machine bound in thirteen problems, and the same schedule in two. Hence for n = 17, and z = 4, there is a difference in algorithms at the $\alpha = .05$ level, with $\hat{\alpha} = .05$.

Experiment G

Table 9 shows that with due dates extended, the job-machine bound produced a less costly schedule in six problems, the cost bound was better in eleven problems, and two schedules were identical. Since n = 17 and z = 6, the hypothesis that the two algorithms are equally efficient cannot be rejected even at the $\alpha = .25$ level. It is also noted that the cost algorithm produced schedules different from the ones produced in experiment B in fifteen of the nineteen problems. The job-machine bound, of course, produced the same schedules as in experiment B.

Discussion of Results

The most important conclusions which can be drawn from the above computational experience are that the cost algorithm in its first pass mode can definitely offer an advantage over time oriented algorithms when economic factors are important, but that this advantage is parameter dependent. There are, however, further points of discussion brought on by the above results.

One aspect which might initially startle the reader is the magnitude of the costs generated. These high costs result from high penalty costs when job completion is delayed several time units past the due date for one or more jobs in the batch. In particular, when

a job is several time units tardy the quadratic contribution to penalty cost can be very high. This should not detract from the results of the analysis however, since experiment F indicates that limiting the penalty cost to a reasonable upper bound does not change the results from a statistical standpoint. In addition, when one considers the time units around which the parameters are structured, the figures seem entirely reasonable since for a twelve job batch being processed over a period of, say 180 days, or half a year, tardiness costs of tens of thousands of dollars are realistic if important jobs are significantly delayed. Of course, the fact that the parameters selected for these experiments were related to days is not important, since relatively speaking the same results would have been obtained if the time units had been minutes or hours instead of days.

There is no observable correlation between best schedule time solutions and best cost solutions. Frequently, in experiments A, B, F, and G the cost bound gives better schedule time solutions than the schedule time algorithm. It is interesting to note from experiment B that a solution with a much longer schedule time will often have a much lower total cost.

Experiment C gives some surprising results. It seems logical that since experiments A and B indicate it is reduction in job tardiness that yields good cost solutions, then an algorithm oriented toward reducing tardiness, like the slack per operation algorithm, should give low cost solutions. In experiment C, however, these results were not achieved. This may be attributed to either of two causes. First, the dispatching rules are inherently local rather than global in effect.

Second, due to the nature of the dispatching technique, it is difficult for even a weighted rule to properly allow for the parametric differences in the economic realm, such as the differences in penalty costs for various jobs in the batch.

Even though experiment F indicates that the efficiency of the cost bound may not be very sensitive to an upper limit on penalty cost, experiment G shows that as due dates give a greater allowance for job completion, the importance of considering penalty cost may decrease. That this should happen is quite apparent from the structure of the problem and from the cost breakdown of experiment B in Table 3. As the due dates approach the range where no jobs are late, the cost algorithm becomes an algorithm bounding on machine idle cost only, which makes it very much like a machine based bound on schedule time. Thus its efficiency might be even less than the efficiency of a "good" schedule time algorithm in situations where due dates are loose or nonexistent.

Further sensitivity to parameters is indicated by the results of experiment E, where all parameters of the problem except machine ordering and processing times were changed to form the replications. It is seen that the desirability of considering economic aspects in the scheduling decision depends heavily on the cost parameters.

As shown in Table 6, the results of experiment D indicate that the computer time required for the cost algorithm is comparable to the time required for the job-machine bound. It would be expected that computation time for even larger problems would increase in an exponential fashion.

Table 2. Experiment A - Comparison of Initial Algorithms

Problem		Initial Job-Machine Bound			Initial Cost Bound		
No.	Size	T(S)	Total Cost	Total Cost After Delay	T(S)	Total Cost	Total Cost After Delay
1	4 X 3	32	4928.66	4923.17	33	4278.72	4274.39*
2	3 X 3	21	1197.02	1195.91*	22	1380.34	1378.76
3	3 X 3	16	398.25	396.52	16	398.25	396.52
4	4 X 3	87	7684.97	7671.69	87	5050.57	5035.36*
5	3 X 2	31	488.92	488.92*	34	674.02	670.62
6	3 X 4	34	3763.07	3760.03	34	3763.07	3760.03
7	6 X 6	58	6534.33	6524.53	69	5179.54	5152.90*
8	8 X 3	153	41345.62	41283.04	156	10538.28	10497.43*
9	8 X 4	189	95669.10	95641.42	213	16376.17	16308.33*
10	10 X 3	155	39799.55	39794.36*	171	59502.54	59477.63
11	10 X 4	170	27296.75	27266.68	211	23102.57	23061.62*
12	12 X 3	205	381333.34	381265.73	190	76318.41	76309.08*
13	12 X 4	233	626170.77	626021.35	235	53611.66	53557.66*
14	8 X 3	137	60882.67	60870.09	134	8055.06	8055.06*
15	8 X 4	167	111644.76	111589.32	165	28866.40	28827.76*
16	10 X 3	207	181199.11	181090.63	194	43044.42	43042.40*
17	10 X 4	215	265516.53	265426.01	220	45120.12	45074.13*
18	12 X 3	205	349766.83	349742.25	218	81177.30	81173.18*
19	12 X 4	259	332331.02	332176.23	246	175902.94	175785.22*

* Indicates lowest cost solution between two algorithms

Table 3. Experiment B- Comparison of Cost Bound and Job-Machine Bound

Problem			Job-Machine Bound				
No.	Size	T(S)	Job Waiting Cost	Machine Idle Cost	Job Penalty Cost	Total Cost	Total Cost After Delay
1	4 X 3	32	42.66	2510.00	2376.00	4928.66	4923.17
2	3 X 3	21	9.02	1020.00	168.00	1197.02	1195.91*
3	3 X 3	16	38.25	360.00	0.00	398.25	396.52
4	4 X 3	87	147.97	3040.00	4497.00	7684.97	7671.69
5	3 X 2	31	88.92	400.00	0.00	488.92	488.92*
6	3 X 4	34	33.07	3730.00	0.00	3763.07	3760.03
7	6 X 6	58	289.33	2870.00	3375.00	6534.33	6524.53*
8	8 X 3	153	976.62	1870.00	38499.00	41345.62	41283.04
9	8 X 4	189	1075.10	9190.00	85404.00	95669.10	95641.42
10	10 X 3	155	508.55	1040.00	38251.00	39799.55	39794.36*
11	10 X 4	170	835.75	6160.00	20301.00	27296.75	27266.68
12	12 X 3	205	1437.34	7520.00	372376.00	381333.34	381265.73
13	12 X 4	233	1903.94	6880.00	612204.00	620987.94	620832.27
14	8 X 3	137	339.67	1720.00	58823.00	60882.67	60870.09
15	8 X 4	167	600.76	4600.00	106444.00	111644.76	111589.32
16	10 X 3	207	835.11	2640.00	177724.00	181199.11	181090.63
17	10 X 4	215	1430.53	15500.00	248586.00	265516.53	265476.01
18	12 X 3	205	591.83	3750.00	345425.00	349766.83	349742.25
19	12 X 4	254	1855.19	5930.00	336219.00	344004.19	343847.85

Table 3. (Continued)

Problem			Cost Bound				
No.	Size	T(S)	Job Waiting Cost	Machine Idle Cost	Job Penalty Cost	Total Cost	Total Cost After Delay
1	4 X 3	30	38.38	2130.00	504.00	2672.38	2670.22*
2	3 X 3	22	6.34	1140.00	234.00	1380.34	1378.76
3	3 X 3	16	38.25	360.00	0.00	398.25	396.52
4	4 X 3	87	138.57	3040.00	1872.00	5050.57	5035.36*
5	3 X 2	34	94.02	580.00	0.00	674.02	670.62
6	3 X 4	34	33.07	3730.00	0.00	3763.07	3760.03
7	6 X 6	67	294.57	3950.00	7656.00	11900.57	11867.48
8	8 X 3	153	675.01	1870.00	6316.00	8861.01	8805.98*
9	8 X 4	213	762.17	12550.00	3064.00	16376.17	16308.33*
10	10 X 3	155	533.78	1040.00	46797.00	48370.78	48368.32
11	10 X 4	209	657.78	15130.00	3054.00	18841.78	18787.26*
12	12 X 3	182	1200.37	2920.00	84362.00	88482.37	88408.72*
13	12 X 4	235	994.17	7080.00	40560.00	48634.17	48542.52*
14	8 X 3	134	313.06	1300.00	6442.00	8055.06	8055.06*
15	8 X 4	191	239.38	7000.00	6222.00	13461.38	13423.72*
16	10 X 3	186	503.70	1170.00	21471.00	23144.70	23131.74*
17	10 X 4	222	748.64	17040.00	22250.00	40038.64	40014.15*
18	12 X 3	218	402.94	5700.00	74429.00	80531.94	80520.43*
19	12 X 4	225	1887.83	3900.00	109141.00	114928.83	114704.76*

* Indicates the lowest cost solution between the two algorithms

Table 4. Costs of Schedules Having Minimal Schedule Time

No.	Size	T(S)*	Total Cost
1	4 X 3	27	3982.54
2	3 X 3	19	878.78
3	3 X 3	16	398.25
4	4 X 3	87	5606.69
5	3 X 2	31	488.92
6	3 X 4	32	3343.02
7	6 X 6	55	5701.92

Table 5. Experiment C - Dispatching Rules

Problem		Slack Per Operation			Weighted Slack Per Operation		
No.	Size	T(S)	Total Cost	Total Cost After Delay	T(S)	Total Cost	Total Cost After Delay
1	4 X 3	35	4657.97	4652.83	33	3415.04	3412.08
2	3 X 3	22	1380.34	1378.76	22	1380.34	1378.76
3	3 X 3	27	1823.76	1820.65	16	398.25	396.52
4	4 X 3	110	21268.03	21238.13	87	8051.00	8037.72
5	3 X 2	42	1177.09	1173.64	36	789.88	788.26
6	3 X 4	51	7512.18	7511.22	44	5858.76	5853.71
7	6 X 6	100	10593.54	10560.87*	82	9147.53	9077.99*
8	8 X 3	216	68777.00	68690.40	144	9471.10	9417.38
9	8 X 4	299	198632.70	198540.74	184	22989.85	22927.68
10	10 X 3	214	153503.28	153460.93	184	69409.17	69399.73
11	10 X 4	319	166949.56	166914.10	195	22267.36	22207.37
12	12 X 3	284	394454.91	394341.37	191	100423.98	100376.82
13	12 X 4	314	350383.50	350077.67	233	81886.96	81794.03
14	8 X 3	228	118993.67	118936.37	159	44507.47	44480.49
15	8 X 4	265	363632.89	363542.68	157	31355.88	31331.01
16	10 X 3	219	140017.24	139911.18	186	50242.19	50203.88
17	10 X 4	239	207480.86	207330.66	215	113016.60	113007.86
18	12 X 3	280	342019.42	341987.41	229	169418.31	169399.27
19	12 X 4	312	399733.27	399481.71	225	207653.04	207306.28

* Indicates a solution better than obtained with the cost bound for the same problem.

Table 6. Experiment D - A Test of Two Large Problems

Problem		Job-Machine Bound				Cost Bound			
No.	Size	T(S)	Total Cost	Total Cost After Delay	Computer Time-Sec.	T(S)	Total Cost	Total Cost After Delay	Computer Time-Sec.
20	20 X 10	177	305091.48	304911.55	195.7	171	101888.23	101796.30	215.7
21	20 X 5	1474	12819198.12	12817962.87	48.3	1522	6121082.87	6119630.75	40.9

Table 7. Experiment E - Variation Within Problems

Problem		Job-Machine Bound			Cost Bound		
No.	Size	T(S)	Total Cost	Total Cost After Delay	T(S)	Total Cost	Total Cost After Delay
4a	4 X 3	87	4911.66	4903.81	131	3635.84	3620.40*
4b	4 X 3	87	5462.70	5455.67*	131	10127.59	10113.38
4c	4 X 3	87	3275.28	3270.31*	100	3911.66	3894.28
4d	4 X 3	87	6074.74	6062.73	100	5521.43	5487.20*
4e	4 X 3	87	10309.22	10295.65	131	9112.79	9086.58*

* Indicates the lowest cost solution between the two algorithms

Table 8. Experiment F - Upper Limit on Job Penalty Cost

Problem		Job-Machine Bound			Cost Bound		
No.	Size	T(S)	Total Cost	Total Cost After Delay	T(S)	Total Cost	Total Cost After Delay
1	4 X 3	32	4928.66	4923.17	30	2672.38	2670.22*
2	3 X 3	21	1197.02	1195.91*	22	1380.34	1378.76
3	3 X 3	16	398.25	396.52	16	398.25	396.52
4	4 X 3	87	7684.97	7671.69	87	5050.57	5035.36*
5	3 X 2	31	488.92	488.92*	34	674.02	670.62
6	3 X 4	34	3763.07	3760.03	34	3763.07	3760.03
7	6 X 6	58	6534.33	6524.53*	67	9156.57	9123.48
8	8 X 3	153	27104.62	27042.04	153	8861.01	8805.98*
9	8 X 4	189	39099.10	39071.42	213	16376.17	16308.33*
10	10 X 3	155	32571.55	32566.36	169	29642.98	29633.64*+
11	10 X 4	170	26820.75	26790.68	209	18841.78	18787.26*
12	12 X 3	205	42985.34	42917.74	182	29485.88	29403.78*+
13	12 X 4	233	59223.94	59068.27	233	37639.93	37477.19*+
14	8 X 3	137	14682.67	14670.09	134	8143.00	8143.00*+
15	8 X 4	167	43038.76	42983.32	191	13461.38	13423.72*+
16	10 X 3	207	48159.11	48050.63	186	23144.70	23131.74*

Table 8. (Continued)

Problem		Job-Machine Bound			Cost Bound		
No.	Size	T(S)	Total Cost	Total Cost After Delay	T(S)	Total Cost	Total Cost After Delay
17	10 X 4	215	65214.53	65124.01	216	35757.36	35740.21*+
18	12 X 3	205	45461.83	45437.25*	213	49086.91	49079.43 +
19	12 X 4	254	70165.19	70008.85	265	58409.22	58303.01*+

* Indicates lowest cost solution between the two algorithms

+ Indicates the cost bound produced a different schedule than the one obtained in experiment B

Table 9. Experiment G - Extended Due Dates

Problem		Job-Machine Bound			Cost Bound		
No.	Size	T(S)	Total Cost	Total Cost After Delay	T(S)	Total Cost	Total Cost After Delay
1	4 X 3	32	2609.47	2603.67*	33	2798.31	2793.79 +
2	3 X 3	21	1092.20	1091.09*	22	1208.31	1207.35
3	3 X 3	16	519.30	517.57	16	519.30	517.57
4	4 X 3	87	3946.58	3914.92*	116	8741.85	8666.62 +
5	3 X 2	31	608.11	608.11*	34	789.07	785.67
6	3 X 4	34	3850.54	3847.51	34	3850.54	3847.51
7	6 X 6	58	3900.01	3889.56*	63	4491.40	4460.43 +
8	8 X 3	153	29924.06	29863.56	179	4582.97	4479.60*+
9	8 X 4	189	26754.60	26725.37	189	11719.92	11655.59*+
10	10 X 3	155	8947.02	8919.35	171	3973.36	3887.05*+
11	10 X 4	170	8996.61	8916.01*	200	19377.68	19224.58 +
12	12 X 3	205	129838.99	129690.04	193	8782.83	8576.84*+
13	12 X 4	233	259330.57	259084.39	230	19564.29	19427.36*+
14	8 X 3	137	8346.95	8334.37	145	4386.57	4365.57*+
15	8 X 4	167	18313.29	18248.62	156	5696.51	5608.83*+
16	10 X 3	207	61247.16	61037.38	202	6157.59	5996.92*+

Table 9. (Continued)

Problem		Job-Machine Bound			Cost Bound		
No.	Size	T(S)	Total Cost	Total Cost After Delay	T(S)	Total Cost	Total Cost After Delay
17	10 X 4	215	60060.45	59941.99	207	19178.82	19037.04*+
18	12 X 3	205	110477.33	110437.11	229	23933.55	23874.41*+
19	12 X 4	254	55347.32	54925.38	239	12007.51	11772.07*+

* Indicates lowest cost solution between two algorithms

+ Indicates the cost bound produced a different schedule than the one obtained in experiment B

CHAPTER IV

SUMMARY AND CONCLUSIONS

The current chapter is organized with the following format. The first section summarizes the research conducted throughout the course of this work. Concluding remarks are then presented. Finally, the last section explores areas fruitful for future research.

Summary of Research

The costs associated with the scheduling decision have been outlined and the pertinent assumptions stated. It has been pointed out that somewhat different formulations for such costs have been developed previously. Following, some of the characteristics of schedules were analyzed in light of the cost formulations. No conditions were found which would indicate an approach to systematically obtain the minimal total cost solution to a scheduling problem

An algorithm was then developed for minimizing the sum of machine idle cost and job penalty cost. The algorithm is heuristic in nature and can be used with or without backtracking, but optimality cannot be guaranteed. The job waiting cost is not considered explicitly in the algorithm until a sequence has been obtained, at which point specific operations are delayed to reduce job waiting costs, hence producing a non-active schedule.

Finally the algorithm in its first pass mode was programmed for use on a digital computer. Several problems from previous published

works were solved using the cost algorithm, an algorithm based on minimizing schedule time, and through the use of a particular dispatching rule. Sensitivity tests were run to investigate the effects of parametric variations on the analysis.

Limitations of the Algorithm

Even with backtracking, a minimal machine idle cost and job penalty cost solution cannot be guaranteed. The job waiting cost is considered only from the standpoint of reducing it after an initial solution is obtained.

The algorithm in its present from is somewhat inflexible. A change in underlying assumptions on the opportunity costs would require modification of the algorithm. It cannot be used when such parameters as job waiting cost rates are functions of time, or when processing costs or times are sequence dependent.

Conclusions

Under the conditions as tested herein a cost algorithm based on a branch and bound procedure renders better scheduling decisions than some of the conventionally accepted procedures. Based on the results of the experimental analysis it is concluded that not only is it inappropriate for a shop manager to disregard opportunity costs in the scheduling decision, but also that is is possible to use an algorithmic process which explicitly considers cost in generating a shop schedule.

On the basis of yet another experiment, it can be concluded that the computational effort required by the cost algorithm is comparable to that required for a similarly structured algorithm used to

minimize schedule time.

Utility of such a cost algorithm is highly dependent on the problem parameters. This implies that when such a procedure is used in an actual shop, care must be taken in considering the validity of assumptions, accuracy of cost estimates, existence of due dates, and similar aspects. For example, if due dates do not exist or are very slack, the scheduler gains little or no advantage in using the cost algorithm developed herein. Another factor to consider is the cost of operating the scheduling system itself, but this consideration is, of course, somewhat standard.

The computational experience presented herein is not extensive. The problem sizes and parameters have been limited in variety. Hence further research as described below is desirable before more substantive claims can be made with regard to the merit of the cost algorithm.

Recommendations for Further Research

In view of the dependence of the cost algorithm on problem parameters, an extensive sensitivity analysis should be undertaken. This sensitivity analysis should be designed to determine such things as (1) the effects of due date setting policy, (2) the range of job waiting cost rates, machine idle cost rates, and job penalty cost rates which make it desirable to consider the respective costs in the scheduling decision, (3) the effects of poor judgement in estimating problem parameters, and (4) the effects of an increase or decrease in the variance of cost rates from machine to machine and from job to job.

It would be highly desirable that further research be performed

in order to investigate the possibility of discovering conditions allowing explicit consideration of the job waiting cost in generating the initial or first pass sequence. If this is not possible, the resequencing of operations to reduce waiting cost after the initial sequence has been obtained might be investigated.

The algorithm given herein could be modified to produce an efficient procedure with backtracking giving minimal possible total machine idle and job penalty cost. Such an algorithm could then be tested in much the same manner as that developed in this research.

The possibility of using dispatching rules to minimize cost was not tested extensively in this work. Further study could be undertaken to investigate the merit of using various dispatching rules to minimize costs in scheduling. Weighted and composite rules, as well as singular rules, could be investigated.

Finally, extensive computational experience is needed, both in the areas developed in this work and in the areas suggested for further research. In particular, an attempt to apply such procedures to an actual job shop would be of interest.

Above all, this research should establish the point that the economic aspects of scheduling have been overly neglected. It is necessary to consider costs, however difficult they are to deal with, if efficient scheduling is to be accomplished. Quoting Conway [17, p. 21], "The costs are nonetheless real, and judging from the demonstrated differences in performance between alternative scheduling procedures there must be some situations in which this choice is vital."

APPENDIX A

JOB-MACHINE BOUND ALGORITHM

With the exception of the lower bound itself, the computational algorithm for the job-machine bound is like the computational algorithm for the cost bound. Hence it is only necessary to present the formulations used in place of formulations (17) through (21) for use in Step 3.2 of the computational algorithm.

The machine based bound is given by:

$$MB_{mn\ell} = \max \left\{ \left[C_{mn\ell} + \sum_{\substack{ij\ell \in \eta \\ ij \neq mn}} t_{ij\ell} \right], \max_{k' \neq \ell} \left[\min_{ijk' \in \eta} (s_{ijk'}) + \sum_{ijk' \in \eta} t_{ijk'} \right] \right\}$$

The job based bound is give by:

$$JB_{mn\ell} = \max_{i} [C_i]$$

where C_i is the earliest completion time of job i, given the particular conflict resolution.

The job-machine based bound is then:

$$B_{mn\ell} = \max[MB_{mn\ell}, JB_{mn\ell}]$$

APPENDIX B

GLOSSARY

Glossary of Terms

1. Active Schedule - a schedule in which it is not possible to decrease the starting time of any operation without the increase in starting time of another operation.

2. Graph - a set of nodes and edges, symbolized as $G(\mathcal{N}, \mathcal{A})$

3. Conjunctive arc - a determinate arc of a graph.

4. Disjunctive arc - an indeterminate arc, showing potential orderings among nodes.

5. Mixed graph - a graph composed of both conjunctive and disjunctive arcs, symbolized $G(\mathcal{N}, \mathcal{C}, \mathcal{D})$.

6. Arborescence - a graph such that there is one arc directed into each node, exclusive of source node.

7. Saturated graph - a graph in which exactly $M(J - 1)$ conjunctive arcs have been selected from the disjunctive set.

8. Unsaturated graph - a graph which is not saturated.

Glossary of Notation Used

i	-	index of jobs, $i = 1, 2, \ldots, J$
j	-	index of operations, $j = 1, 2, \ldots, g_i$
k	-	index of machines, $k = 1, 2, \ldots, M$
r	-	job waiting cost rate
z	-	test statistic

a_i - penalty cost coefficient

d_i - due date of job i

r_k - machine idle cost rate, machine k

s_{ijk} - start time of operation ijk

t_{ijk} - processing time of operation ijk

v_{ij} - value added to job i in operation j

y_{ij} - waiting time of job i before operation j

(ijk) - operation ijk, or node ijk

(k, ℓ) - arc directed from node k to node ℓ

C_i - completion time, job i

C^k - completion time of last operation on machine k

C_{ijk} - completion time of operation ijk

I_{ijk} - idle time on machine k before processing ijk

S_ℓ - schedule ℓ

$T(S)$ - schedule time

$TC(S)$ - total variable cost due to schedule S

T'_i - tardiness of job i

V_{ij} - value of job i before operation j

$\mathcal{A}$ - set of arcs of a graph

$\mathcal{C}$ - set of conjunctive arcs of a graph

$\mathcal{D}$ - set of disjunctive arcs of a graph

$\mathcal{N}$ - set of nodes of a graph

η - set of unscheduled operations

$\beta(S)$ - job waiting cost component

$\gamma(S)$ - machine idle cost component

$\delta(S)$ - job penalty cost component

σ - set of singletons

$\emptyset$ - the empty set

Ψ_p - candidate set at iteration p

θ - a conflict set

APPENDIX C

ALGORITHM SOURCE LISTING

```
00100      1*      C
00100      2*      C     THIS ROUTINE USES A BRANCH AND BOUND ON COMPOSITE MACHINE
00100      3*      C     IDLE COST AND JOB LATENESS COST TO SOLVE THE SHOP SCHEDULING
00100      4*      C     PROBLEM.
00100      5*      C     AN OPTION IS AVAILABLE TO USE A COMPOSITE JOB-MACHINE BASED BOUND
00100      6*      C     INSTEAD OF THE BOUND ON COST
00100      7*      C
00100      8*      C
00100      9*      C   THE FOLLOWING VARIABLES OR PARAMETERS ARE OF INTEREST
00100     10*      C
00100     11*      C   IPROB     SPECIFIES THE NUMBER OF PROBLEMS TO BE RUN
00100     12*      C   JOBS      JOB SIZE
00100     13*      C   MACHS     MACHINE SIZE
00100     14*      C   IG        MACHINE ORDERING MATRIX
00100     15*      C   IT        PROCESSING TIME MATRIX
00100     16*      C   MRATE     MACHINE IDLE COST MATRIX
00100     17*      C   IDATE     JOB DUE DATE MATRIX
00100     18*      C   JPA       LINEAR JOB PENALTY RATE MATRIX
00100     19*      C   JPB       QUADRATIC JOB PENALTY RATE MATRIX
00100     20*      C   RATE      JOB WAITING RATE
00100     21*      C   IVALUE    CUMULATIVE JOB VALUE MATRIX
00100     22*      C   MTSUM     TOTAL PROCESSING TIME FOR A MACHINE
00100     23*      C   IQ,IA     MATRIX PAIR DESIGNATING CURRENT ARCS.  IQ-ORIGINATING NODE,
00100     24*      C             IA-TERMINATING NODE
00100     25*      C   J         MACHINE BASED NODE SETS
00100     26*      C   IU        JOB AVAILABILITY INDEX
00100     27*      C   IV        MACHINE AVAILABILITY INDEX
00100     28*      C   ICAN      CANDIDATE SET MATRIX
00100     29*      C   NFIX      0   OPERATION NOT YET SCHEDULED
00100     30*      C             1   OPERATION SCHEDULED
00100     31*      C   MBJB      0   CONFLICT RESOLUTION BASED ON COST
00100     32*      C             1   CONFLICT RESOLUTION BASED ON COMPOSITE MACHINE
00100     33*      C                 AND JOB BASED BOUND
00100     34*      C   IPRIN     0   PRINT ALL INTERMEDIATE INFORMATION
00100     35*      C             1   PRINT ONLY ITERATIVE SUMMARIES
00100     36*      C   JLIM      0   NO LIMIT ON PENALTY COST
00100     37*      C             1   PENALTY COST LIMITED BY JOB VALUE
00100     38*      C   IDELAY    0   DO NOT USE DELAY SCHEME
00100     39*      C             1   USE DELAY SCHEME TO REDUCE JOB WAITING COST
00100     40*      C
00101     41*            INTEGER D
00103     42*            DIMENSION IA(200),IQ(200),IG(200),IGP(200),IT(200),IU(200),IV(200)
00103     43*           1,D(200),J(10,30),IQD(200),IAD(200),MJ(200),NFIX(200),ICAN(30),IVAL
00103     44*           2UE(30,11),MRATE(10),JPA(30),JPB(30),IDATE(30),MTSUM(10),NPIE(200),
00103     45*           3ISTR(200),ICOM(200)
00104     46*            MBJB=0
00105     47*            IPRIN=0
00106     48*            IPROB=1
00107     49*            JLIM=0
00110     50*            LIMV=2
00111     51*            IDELAY=1
00112     52*            IGO=0
00113     53*       8829 IGO=IGO+1
00114     54*       9772 FORMAT(' ','CONFLICTS RESOLVED ON COST BOUND')
00115     55*       9773 FORMAT(' ','CONFLICTS RESOLVED ON JOB-MACH BOUND')
00116     56*            IF(MBJB)9770,9770,9771
00121     57*       9770 WRITE(6,9772)
00123     58*            GO TO 9775
00124     59*       9771 WRITE(6,9773)
00126     60*       9775 ITER=0
00126     61*      C
00126     62*      C     READ PROBLEM PARAMETERS
00126     63*      C
00127     64*            READ(5,1) JOBS,MACHS
00133     65*          1 FORMAT(1615)
00134     66*       9521 FORMAT(1X,16I5)
00135     67*       9655 FORMAT(F10.5)
00136     68*            JM=JOBS*MACHS
00137     69*            L=1
00140     70*            IX=1
00141     71*            KX=MACHS
00142     72*       9524 FORMAT(' ',' THE MACHINE ORDERING MATRIX IS')
00143     73*            WRITE(6,9524)
00145     74*          2 CONTINUE
00146     75*            READ(5,1) (IG(I),I=IX,KY)
00154     76*            WRITE(6,1) (IG(I),I=IX,KX)
00162     77*            IF(L.EQ.JOBS) GO TO 4
00164     78*            L=L+1
00165     79*            IX=IX+MACHS
00166     80*            KX=KY+MACHS
00167     81*            GO TO 2
```

```
00170     82*        4 L=1
00171     83*          IX=1
00172     84*          KX=MACHS
          [illegible]  5 CONTINUE
00173     85*          READ(5,1)(IT(I),I=IX,KX)
00201     86*          IF(L.EQ.JOBS) GO TO 6
00203     87*          L=L+1
00204     88*          IX=IX+MACHS
00205     89*          KX=KX+MACHS
00206     90*          GO TO 5
00207     91*        6 CONTINUE
00210     92*          DO 7 I=1,JM
00213     93*          IGP(I)=I
00214     94*        7 CONTINUE
00216     95*          WRITE(6,12)
00220     96*       12 FORMAT(' ','THE NODE SETS ARE')
00221     97*          IX=1
00222     98*          KX=MACHS
00223     99*       10 CONTINUE
00224    100*          WRITE(6,1)(IGP(I),I=IX,KX)
00232    101*          IF(KX.EQ.JM) GO TO 11
00234    102*          IX=IX+MACHS
00235    103*          KX=KX+MACHS
00236    104*          GO TO 10
00237    105*       11 CONTINUE
00240    106*          WRITE(6,13)
00242    107*       13 FORMAT(' ','THE PROCESSING TIME MATRIX IS')
00243    108*          IX=1
00244    109*          KX=MACHS
00245    110*       14 CONTINUE
00246    111*          WRITE(6,1)(IT(I),I=IX,KX)
00254    112*          IF(KX.EQ.JM) GO TO 15
00256    113*          IX=IX+MACHS
00257    114*          KX=KX+MACHS
00260    115*          GO TO 14
00261    116*       15 CONTINUE
00262    117*          READ(5,1) (MRATE(I),I=1,MACHS)
00270    118*          READ(5,1) (IDATE(I),I=1,JOBS)
00276    119*          READ(5,1) (JPA(I),I=1,JOBS)
00304    120*          READ(5,1) (JPB(I),I=1,JOBS)
00312    121*          READ(5,9655) RATE
00315    122*    9522  FORMAT(' ','MRATE, IDATE, JPA, JPB MATRICES')
00316    123*          WRITE(6,9522)
00320    124*          WRITE(6,1) (MRATE(I),I=1,MACHS)
00326    125*          WRITE(6,1) (IDATE(I),I=1,JOBS)
00334    126*          WRITE(6,1) (JPA(I),I=1,JOBS)
00342    127*          WRITE(6,1) (JPB(I),I=1,JOBS)
00350    128*    9523  FORMAT(' ','RATE =',F10.5)
00351    129*          WRITE(6,9523) RATE
00354    130*          MAK=MACHS+1
00355    131*          WRITE(6,9656)
00357    132*    9656  FORMAT(' ','THE JOB VALUE MATRIX IS')
00360    133*          DO 9641 IJ=1,JOBS
00363    134*          READ(5,1) (IVALUE(IJ,IK),IK=1,MAK)
00371    135*          WRITE(6,1) (IVALUE(IJ,IK),IK=1,MAK)
00377    136*    9641  CONTINUE
00377    137*    C
00377    138*    C     COMPUTE THE MACHINE BASED NODE SETS.
00377    139*    C
00401    140*          DO 5727 IGB=1,MACHS
00404    141*          DO 5723 IX=1,JOBS
00407    142*          J(IGB,IX)=0
00410    143*    5723  CONTINUE
00412    144*    5727  CONTINUE
00414    145*          DO 200 IX=1,JM
00417    146*          MJ(IX)=0
00420    147*          D(IX)=0
00421    148*          IU(IX)=0
00422    149*          IV(IX)=0
00423    150*     200  CONTINUE
00425    151*          ICOUN=1
00426    152*          IKOUN=MACHS
00427    153*          DO 100 IX=1,JM
00432    154*          IP=IG(IX)
00433    155*          IPX=IP/100
00434    156*          IPQ=IP-(IPX*100)
00435    157*          KQ=JOBS*(IPQ-1)+ICOUN
00436    158*          MJ(KQ)=IX
00437    159*          IF(IX.EQ.IKOUN) GO TO 102
00441    160*          GO TO 100
00442    161*      102 IF(IX.EQ.JM) GO TO 103
```

```
00444   162*          ICOUN=ICOUN+1
00445   163*          IKOUN=IKOUN+MACHS
00446   164*          GO TO 100
00447   165*      103 CONTINUE
00450   166*      100 CONTINUE
00452   167*          IBGN=1
00453   168*          IEND=JOBS
00454   169*          DO 5921 IB=1,MACHS
00457   170*          IXI=0
00460   171*          DO 5923 IX=IBGN,IEND
00463   172*          IXI=IXI+1
00464   173*          JJ(IB,IXI)=MJ(IX)
00465   174*      5923 CONTINUE
00467   175*          IF(IEND.EQ.JM) GO TO 5921
00471   176*          IBGN=IBGN+JOBS
00472   177*          IEND=IEND+JOBS
00473   178*      5921 CONTINUE
00475   179*          IX=1
00476   180*          KX=JOBS
00477   181*       151 CONTINUE
00500   182*          IKK=1
00501   183*       500 FORMAT(' ',' THE MACHINE BASED NODE SETS ARE')
00502   184*          WRITE(6,500)
00504   185*      1511 WRITE(6,1)(JJ(IKK,ID),ID=IX,KX)
00512   186*          IF(IKK.EQ.MACHS) GO TO 152
00514   187*          IKK=IKK+1
00515   188*          GO TO 1511
00516   189*       152 CONTINUE
00517   190*          KX=(JOBS-1)*MACHS
00520   191*          DO 201 I=1,KX
00523   192*          IQ(I)=0
00524   193*          IA(I)=0
00525   194*          IAD(I)=0
00526   195*          IQD(I)=0
00527   196*       201 CONTINUE
00531   197*          DO 9520 IJ=1,MACHS
00534   198*          MTSUM(IJ)=0
00535   199*          DO 9510 IK=1,JOBS
00540   200*          I=JJ(IJ,IK)
00541   201*          MTSUM(IJ)=MTSUM(IJ)+IT(I)
00542   202*       9510 CONTINUE
00544   203*       9520 CONTINUE
00544   204*      C
00544   205*      C             COMPUTE THE JOB AVAILABILITY TABLES - IU.
00544   206*      C
00546   207*          IX=1
00547   208*          KX=MACHS
00550   209*          IKQ=KX-1
00551   210*        192 CONTINUE
00552   211*          DO 180 I=IX,KX
00555   212*          IDD=I-1
00556   213*          IKQQ=KX-IKQ
00557   214*          IF(I.EQ.IKQQ) GO TO 181
00561   215*          IU(I)=IU(IDD)+IT(IDD)
00562   216*          GO TO 180
00563   217*       181 IU(I)=0
00564   218*       180 CONTINUE
00566   219*          IF(KX.EQ.JM) GO TO 193
00570   220*          IX=IX+MACHS
00571   221*          KX=KX+MACHS
00572   222*          GO TO 192
00573   223*       193 CONTINUE
00574   224*          WRITE(6,7008)
00576   225*       7008 FORMAT(' ',' THE MACHINE START TIMES ARE')
00577   226*          WRITE(6,1)(IU(I),I=1,JM)
00605   227*          ITEX=0
00606   228*          IETX=0
00607   229*       3392 IETX=MACHS+IETX
00610   230*          IKEX=IU(IETX)+IT(IETX)
00611   231*          IF(IKEX.GT.ITEX) ITEX=IKEX
00613   232*          IF(IETX.NE.JM) GO TO 3392
00615   233*          ISTEP=ITEX
00616   234*          DO 950 I=1,JM
00621   235*          NFIX(I)=0
00622   236*       950 CONTINUE
00622   237*      C
00622   238*      C     START OF NEW ITERATION
00622   239*      C
00624   240*       1000 ITER=ITER+1
00625   241*          WRITE(6,9642) ITER
00630   242*       9642 FORMAT(' ',' NFIX AT ITERATION',I3)
```

```
00031   243*          WRITE(6,1)(NFIX(I),I=1,JM)
00037   244*          DO 951 IJK=1,JOBS
00042   245*          ICAN(IJK)=0
00043   246*   951 CONTINUE
00045   247*  4000 CONTINUE
00045   248*     C
00045   249*     C      SCAN FOR CANDIDATES.
00045   250*     C
00046   251*          IF(IPRIN.EQ.1) GO TO 4040
00050   252*          WRITE(6,2000)
00052   253*  2000 FORMAT(' ',' THE INITIAL SET OF CANDIDATES ARE')
00053   254*  4040 CONTINUE
00054   255*          IM=0
00055   256*  1001 IM=IM+1
00056   257*          ISTIC=0
00057   258*          IEND=IM*MACHS
00060   259*          IST=(IM-1)*MACHS+1
00061   260*          DO 1002 IOUT=IST,IEND
00064   261*          IF(ISTIC.NE.0) GO TO 1002
00066   262*          IF(NFIX(IOUT).EQ.0) GO TO 1003
00070   263*          GO TO 1002
00071   264*  1003 ISTIC=IOUT
00072   265*          IF(IPRIN.EQ.1) GO TO 4041
00074   266*          WRITE(6,1) ISTIC
00077   267*  4041 CONTINUE
00100   268*  1002 CONTINUE
00102   269*          ICAN(IM)=ISTIC
00103   270*          IF(IEND.EQ.JM) GO TO 1006
00105   271*          GO TO 1001
00106   272*  1006 CONTINUE
00107   273*          K=(JOBS-1)*MACHS
00107   274*     C
00107   275*     C      RESOLVE CONFLICTS AND SCHEDULE OPERATIONS
00107   276*     C
00110   277*          CALL BREAK(IG,ICAN,IQ,IA,IQD,IAD,JOBS,MACHS,JM,K,J,IU,IV,IT,ISTER,
00110   278*         1ITER,NFIX,MTSUM,MRATE,IDATE,JPA,JPB,MBJR,IVALUE,JLIM)
00111   279*  9650 DO 9651 I=1,JM
00114   280*          IF(NFIX(I).EQ.0) GO TO 9652
00116   281*  9651 CONTINUE
00120   282*          GO TO 3000
00121   283*  9652 IF(IPRIN.EQ.1) GO TO 1000
00123   284*          WRITE(6,4619) ITER
00126   285*  4619 FORMAT('    ',' THE UPDATE AT ITERATION',I3,'IS')
00127   286*          DO 4718 IP=1,K
00132   287*          WRITE(6,1) IQ(IP),IA(IP)
00136   288*  4718 CONTINUE
00136   289*     C
00136   290*     C      END OF ITERATION
00136   291*     C
00140   292*          GO TO 1000
00140   293*     C
00140   294*     C      PRINT THE SCHEDULE TIME AND ARCS OF THE GRAPH
00140   295*     C
00141   296*  3000 CONTINUE
00142   297*          WRITE(6,3001) ISTER
00145   298*  3001 FORMAT('  ',' THE SCHEDULE TIME CAN BE GIVEN AS',I5)
00146   299*          DO 3010 IP=1,K
00151   300*          WRITE(6,1) IQ(IP),IA(IP)
00155   301*  3010 CONTINUE
00155   302*     C
00155   303*     C   TOTAL COST CALCULATION
00155   304*     C
00157   305*          IWVT=0
00160   306*          MAK=MACHS+1
00161   307*          I=1
00162   308*          DO 9620 IJ=1,JOBS
00165   309*          DO 9617 IK=1,MAK
00170   310*          IF(IV.EQ.1) GO TO 9602
00172   311*          IZ=I-1
00173   312*          IF(IU(IZ).GT.IV(IZ)) GO TO 9605
00175   313*          ICL=IV(IZ)+IT(IZ)
00176   314*          GO TO 9608
00177   315*  9605 ICL=IU(IZ)+IT(IZ)
01000   316*          GO TO 9608
01001   317*  9602 ICL=0
01002   318*  9608 IF(IK.EQ.MAK) GO TO 9611
01004   319*          IF(IU(I).GT.IV(I)) GO TO 9610
01006   320*          IST=IV(I)
01007   321*          GO TO 9612
01010   322*  9610 IST=IU(I)
01011   323*          GO TO 9612
```

```
01012   324*      9611 IST=IDATE(IJ)
01013   325*      9612 IWAIT=IST-ICL
01014   326*           IF(IWAIT.LE.0) IWAIT=0
01016   327*           IWVT=IWVT+(IWAIT*IVALUE(IJ,IK))
01017   328*           I=I+1
01020   329*      9617 CONTINUE
01022   330*           I=I-1
01023   331*      9620 CONTINUE
01025   332*           WVT=IWVT
01026   333*           BETA=RATE*WVT
01027   334*           NBETA=BETA+0.5
01030   335*           NGAMMA=0
01031   336*           DO 9625 K=1,MACHS
01034   337*           NGAMMA=NGAMMA+(MRATE(K)*(ISTER-MTSUM(K)))
01035   338*      9625 CONTINUE
01037   339*           NDELTA=0
01040   340*           KX=MACHS
01041   341*           DO 9633 I=1,JOBS
01044   342*           IF(IU(KX).GT.IV(KX)) GO TO 9626
01046   343*           JCOMP=IV(KX)+IT(KX)
01047   344*           GO TO 9628
01050   345*      9626 JCOMP=IU(KX)+IT(KX)
01051   346*      9628 ITR=JCOMP-IDATE(I)
01052   347*           IF(ITR.GT.0) GO TO 9630
01054   348*           ITARD=0
01055   349*           GO TO 9631
01056   350*      9630 ITARD=ITR
01057   351*      9631 JPEN=JPA(I)*ITARD+JPB(I)*(ITARD**2)
01060   352*           IF(JLIM.EQ.0) GO TO 9830
01062   353*           LIMT=LIMV*IVALUE(I,MAK)
01063   354*           IF(LIMT.LT.JPEN) JPEN=LIMT
01065   355*      9830 NDELTA=NDELTA+JPEN
01066   356*           KX=KX+MACHS
01067   357*      9633 CONTINUE
01071   358*           GAMMA=NGAMMA
01072   359*           DELTA=NDELTA
01073   360*           COST=BETA+GAMMA+DELTA
01074   361*           NCOST=NBETA+NGAMMA+NDELTA
01075   362*           WRITE(6,9635) BETA
01100   363*           WRITE(6,9636) GAMMA
01103   364*           WRITE(6,9637) DELTA
01106   365*           WRITE(6,9638) COST
01111   366*      9635 FORMAT('  ','BETA  $',F15.2)
01112   367*      9636 FORMAT('  ','GAMMA $',F15.2)
01113   368*      9637 FORMAT('  ','DELTA $',F15.2)
01114   369*      9638 FORMAT('  ','COST  $',F15.2)
01114   370*      C
01114   371*      C         DELAY OF OPERATIONS TO REDUCE JOB WAITING COST - NBETA
01114   372*      C
01115   373*           IF(IDELAY.EQ.0) GO TO 9700
01117   374*           DO 9705 I=1,JM
01122   375*           NPIE(I)=0
01123   376*           IF(IU(I).GT.IV(I)) GO TO 9703
01125   377*           ISTR(I)=IV(I)
01126   378*           GO TO 9704
01127   379*      9703 ISTR(I)=IU(I)
01130   380*      9704 ICOM(I)=ISTR(I)+IT(I)
01131   381*      9705 CONTINUE
01133   382*      9706 IDD=0
01134   383*           DO 9710 I=1,JM
01137   384*           IF(NPIE(I).EQ.1) GO TO 9710
01141   385*           IF(ICOM(I).LE.IDD) GO TO 9710
01143   386*           IDD=ICOM(I)
01144   387*           IZ=I
01145   388*      9710 CONTINUE
01147   389*           IP=IG(IZ)
01150   390*           IPX=IP/100
01151   391*           IPQ=IP-(IPX*100)
01152   392*           DO 9715 KX=MACHS,JM,MACHS
01155   393*           IF(IZ.EQ.KX) GO TO 9716
01157   394*      9715 CONTINUE
01161   395*           GO TO 9725
01162   396*      9716 IF(IDD.GE.IDATE(IPX)) GO TO 9732
01164   397*           ISMIN=IDATE(IPX)
01165   398*           DO 9720 IJ=1,JOBS
01170   399*           IPQQ=J(IPQ,IJ)
01171   400*           IF(NPIE(IPQQ).EQ.0) GO TO 9720
01173   401*           IF(ISTR(IPQQ).LT.ISMIN) ISMIN=ISTR(IPQQ)
01175   402*      9720 CONTINUE
01177   403*           IF(ISTER.LT.ISMIN) ISMIN=ISTER
01201   404*           ICOM(IZ)=ISMIN
```

```
01202    405*            GO TO 9732
01203    406*        9725 MAK=IZ+1
01204    407*             ISMIN=ISTR(MAK)
01205    408*             DO 9730 IJ=1,JOBS
01210    409*             IPQQ=J(IPQ,IJ)
01211    410*             IF(NPIE(IPQQ).EQ.0) GO TO 9730
01213    411*             IF(ISTR(IPQQ).LT.ISMIN) ISMIN=ISTR(IPQQ)
01215    412*        9730 CONTINUE
01217    413*             ICOM(IZ)=ISMIN
01220    414*        9732 NPIE(IZ)=1
01221    415*             ISTR(IZ)=ICOM(IZ)-IT(IZ)
01222    416*             DO 9735 I=1,JM
01225    417*             IF(NPIE(I).EQ.0) GO TO 9706
01227    418*        9735 CONTINUE
01231    419*        9738 FORMAT(' ','START TIME MATRIX AFTER DELAY')
01232    420*        9739 FORMAT(' ','COMPLETION TIME MATRIX AFTER DELAY')
01233    421*             WRITE(6,9738)
01235    422*             WRITE(6,9521) (ISTR(I),I=1,JM)
01243    423*             WRITE(6,9739)
01245    424*             WRITE(6,9521) (ICOM(I),I=1,JM)
01245    425*        C
01245    426*        C         CALCULATION OF WAITING COST AFTER DELAY
01245    427*        C
01253    428*         9740 IWVT=0
01254    429*              MAK=MACHS+1
01255    430*              I=1
01256    431*              DO 9752 IJ=1,JOBS
01261    432*              DO 9750 IK=1,MAK
01264    433*              IF(IV.EQ.1) GO TO 9742
01266    434*              IZ=I-1
01267    435*              IF(IK.EQ.MAK) GO TO 9743
01271    436*              IWAIT=ISTR(I)-ICOM(IZ)
01272    437*              GO TO 9745
01273    438*         9742 IWAIT=ISTR(I)
01274    439*              GO TO 9745
01275    440*         9743 IWAIT=IDATE(IJ)-ICOM(IZ)
01276    441*              IF(IWAIT.LT.0) IWAIT=0
01300    442*         9745 IWV=IWAIT*IVALUE(IJ,IK)
01301    443*              IWVT=IWVT+IWV
01302    444*              I=I+1
01303    445*         9750 CONTINUE
01305    446*              I=I-1
01306    447*         9752 CONTINUE
01310    448*              WVT=IWVT
01311    449*              BETA=RATE*WVT
01312    450*              NBETA=BETA+0.5
01313    451*              COST=BETA+GAMMA+DELTA
01314    452*              NCOST=NBETA+NGAMMA+NDELTA
01315    453*         9755 FORMAT(' ','COSTS AFTER DELAY')
01316    454*              WRITE(6,9755)
01320    455*              WRITE(6,9635) BETA
01323    456*              WRITE(6,9638) COST
01326    457*         9700 CONTINUE
01327    458*         9760 FORMAT(/////)
01330    459*              WRITE(6,9760)
01332    460*              IF(IGO.LT.IPROB) GO TO 8829
01334    461*              STOP
01335    462*              END


00101      1*              SUBROUTINE BREAK(IG,ICAN,IQ,IA,IQD,IAD,JOBS,MACHS,JM,K,J,IU,IV,IT,
00101      2*             1ISTEP,ITER,NFIX,MTSUM,MPATE,IDATE,JPA,JPB,MBJB,IVALUE,JLIM)
00101      3*        C
00101      4*        C
00101      5*        C
00101      6*        C     THIS ROUTINE IDENTIFIES CANDIDATES FOR SEQUENCING, RESOLVES
00101      7*        C     CONFLICTS, AND UPDATES THE CURRENT GRAPH.
00101      8*        C
00101      9*        C
00101     10*        C
00101     11*        C   THE FOLLOWING VARIABLES ARE OF INTEREST
00101     12*        C   ISET      MATRIX OF SINGLETONS FROM THE CANDIDATE SET
00101     13*        C   IWORK     A CONFLICT SET
00101     14*        C   IINI      SET OF OPERATIONS TO BE SCHEDULED AT THIS ITERATION
00101     15*        C   IHOLD     SET OF NODES HELD OVER
```

```
00101     16*      C
00101     17*      C
00103     18*            DIMENSION IG(200),ICAN(30),IQ(200),IA(200),IQ)(200),IAD(200),ISET(
00103     19*           130),IHULD(30),IWORK(30),NFIX(200),IIN1(30),IU(200),IV(200),IT(200)
00103     20*           2,J(10,30),MRATE(10),JPA(30),JPB(30),IDATE(30),MTSUM(10),IVALUE(30,
00103     21*           311)
00103     22*      C
00104     23*            IBDX=0
00105     24*          1 FORMAT(16I5)
00106     25*            IPRIN=0
00107     26*            IBUM=0
00110     27*            DO 9227 IKIK=1,JOBS
00113     28*            IF(ICAN(IKIK).NE.0) IBUM=IBUM+1
00115     29*       9227 CONTINUE
00117     30*            DO 2000 IBUL=1,JOBS
00122     31*            ISET(IBUL)=0
00123     32*            IHULD(IBUL)=0
00124     33*       2000 CONTINUE
00124     34*      C
00124     35*      C     IDENTIFY SINGLETONS
00124     36*      C
00126     37*            DO 1000 IKOH=1,JOBS
00131     38*            IF(ICAN(IKOH).EQ.0) GO TO 1000
00133     39*            IMIN=0
00134     40*            IEE=ICAN(IKOH)
00135     41*            IWW=IG(IEE)/100
00136     42*            IWWW=IG(IEE)-IWW*100
00137     43*            DO 1001 IKOI=1,JOBS
00142     44*            IF(ICAN(IKOI).EQ.0) GO TO 1001
00144     45*            IF(IKOI.EQ.IKOH) GO TO 1001
00146     46*            IEX=ICAN(IKOI)
00147     47*            IWP=IG(IEX)
00150     48*            IWWP=IWP-(IWP/100)*100
00151     49*            IF(IWWP.EQ.IWWW) IMIN=IMIN+1
00153     50*       1001 CONTINUE
00155     51*            IF(IMIN.GT.0) GO TO 1000
00157     52*            IF(IPRIN.EQ.1) GO TO 4040
00161     53*            WRITE(6,1007) ICAN(IKOH)
00164     54*       1007 FORMAT('  ','A SINGLETON IS',I10)
00165     55*       4040 CONTINUE
00166     56*            DO 1010 IBUL=1,IBUM
00171     57*            IF(ISET(IBUL).EQ.0) GO TO 1011
00173     58*       1010 CONTINUE
00175     59*       1011 ISET(IBUL)=ICAN(IKOH)
00176     60*       1000 CONTINUE
00200     61*            IF(ISET(1).GT.0.AND.ITER.EQ.1) GO TO 5001
00202     62*            IF(ITER.EQ.1) GO TO 4618
00202     63*      C
00202     64*      C     CHECK THE SINGLETONS FOR HOLDOVER NODES.
00202     65*      C
00204     66*            INON=0
00205     67*            DO 5000 IBUL=1,IBUM
00210     68*            IF(ISET(IBUL).NE.0) INON=INON+1
00212     69*       5000 CONTINUE
00214     70*            IF(INON.EQ.JOBS) GO TO 5001
00216     71*            DO 5050 IBUL=1,JOBS
00221     72*            IF(ISET(IBUL).EQ.0) GO TO 5050
00223     73*            IEE=ISET(IBUL)
00224     74*            IWX=IG(IEE)/100
00225     75*            IWWX=IG(IEE)-IWX*100
00226     76*            DO 5060 INIX=1,JOBS
00231     77*            IF(ICAN(INIX).EQ.ISET(IBUL)) GO TO 5060
00233     78*            IMYB=ICAN(INIX)+1
00234     79*            IF(IMYB.GT.JM) GO TO 5060
00236     80*            IF(NFIX(IMYB).EQ.1) GO TO 5060
00240     81*            IWC=IG(IMYB)/100
00241     82*            IWWC=IG(IMYB)-IWC*100
00242     83*            IF(IWWC.EQ.IWWX) IHULD(IBUL)=ISET(IBUL)
00244     84*       5060 CONTINUE
00246     85*       5050 CONTINUE
00250     86*            ISCON=0
00251     87*            IRCON=0
00252     88*            DO 5418 I100=1,JOBS
00255     89*            IF(ICAN(I100).NE.0) IRCON=IRCON+1
00257     90*            IF(IHULD(I100).NE.0) ISCON=ISCON+1
```

```
00261     91*      3418 CONTINUE
00263     92*           IF(IPCON.EQ.ISCON) GO TO 4618
00265     93*           IF(IPRIN.EQ.1) GO TO 4041
00267     94*           WRITE(6,9050)
00271     95*      9050 FORMAT(' ','HOLD THE FOLLOWING NODES')
00272     96*           WRITE(6,1)(IHOLD(IQ0),IQ0=1,JOBS)
00300     97*      4041 CONTINUE
00300     98*      C
00300     99*      C     UPDATE THE CANDIDATE SET.
00300    100*      C
00301    101*           DO 5061 IQ01=1,JOBS
00304    102*           IMET=IHOLD(IQ01)
00305    103*           DO 5062 IQ02=1,JOBS
00310    104*           IF(ICAN(IQ02).EQ.IMET) ICAN(IQ02)=0
00312    105*      5062 CONTINUE
00314    106*      5061 CONTINUE
00316    107*      4618 CONTINUE
00317    108*           IF(IPRIN.EQ.1) GO TO 4042
00321    109*           WRITE(6,6092)
00323    110*      6092 FORMAT(' ','THE FINAL CANDIDATE SET IS')
00324    111*           WRITE(6,1) (ICAN(IQ02),IQ02=1,JOBS)
00332    112*      4042 CONTINUE
00332    113*      C
00332    114*      C     SCHEDULING OF SINGLETONS
00332    115*      C
00333    116*      9670 IIND=0
00334    117*           DO 9671 I=1,JOBS
00337    118*           IIN1(I)=0
00340    119*      9671 CONTINUE
00342    120*           DO 9680 IQB=1,JOBS
00345    121*           DO 9678 I=1,JOBS
00350    122*           IF(ICAN(I).NE.ISET(IQB)) GO TO 9678
00352    123*           IF(ICAN(I).EQ.0) GO TO 9678
00354    124*           IIND=1
00355    125*           ISLT=ISET(IQB)
00356    126*           DO 9675 ITBB=1,JOBS
00361    127*           IF(IIN1(ITBB).EQ.0) GO TO 9677
00363    128*      9675 CONTINUE
00365    129*      9677 IIN1(ITBB)=ISLT
00366    130*           ICAN(I)=0
00367    131*      9678 CONTINUE
00371    132*      9680 CONTINUE
00373    133*           IF(IIND.EQ.0) GO TO 9699
00375    134*           IF(IPRIN.EQ.1) GO TO 4044
00377    135*      9681 FORMAT(' ','SINGLETONS SCHEDULED BEFORE RESOLVING CONFLICTS')
00400    136*           WRITE(6,9681)
00402    137*           WRITE(6,1) (IIN1(I),I=1,JOBS)
00410    138*      9691 FORMAT(' ','THE NEW CANDIDATE SET IS')
00411    139*           WRITE(6,9691)
00413    140*           WRITE(6,1) (ICAN(I),I=1,JOBS)
00421    141*           GO TO 4044
00422    142*      9699 CONTINUE
00423    143*           IIND=0
00423    144*      C
00423    145*      C     IDENTIFY CONFLICTS.
00423    146*      C
00424    147*           DO 8080 ITB=1,JOBS
00427    148*           IWORK(ITB)=0
00430    149*           IIN1(ITB)=0
00431    150*      8080 CONTINUE
00433    151*           ITB=1
00434    152*           DO 6040 IBUL=1,JOBS
00437    153*           IF(ICAN(IBUL).EQ.0) GO TO 6040
00441    154*           IEE=ICAN(IBUL)
00442    155*           IWW=IG(IEE)/100
00443    156*           IWWW=IG(IEE)-IWW*100
00444    157*           DO 6041 IBULL=IBUL,JOBS
00447    158*           IF(ICAN(IBULL).EQ.ICAN(IBUL)) GO TO 6041
00451    159*           IF(ICAN(IBULL).EQ.0) GO TO 6041
00453    160*           IEX=ICAN(IBULL)
00454    161*           IWX=IG(IEX)/100
00455    162*           IWWX=IG(IEX)-IWX*100
00456    163*           IF(IWWX.EQ.IWWW) GO TO 6042
00460    164*           GO TO 6041
00461    165*      6042 IWORK(1)=ICAN(IBUL)
00462    166*           ITB=ITB+1
00463    167*           IWORK(ITB)=ICAN(IBULL)
00464    168*      6041 CONTINUE
00466    169*           IF(IPRIN.EQ.1) GO TO 4043
00470    170*      9759 FORMAT(' ','A CONFLICT SET IS')
```

```
00471    171*          WRITE(6,9759)
00473    172*          WRITE(6,1)(IWORK(IZI),IZI=1,JOBS)
00501    173*     4043 CONTINUE
00502    174*          IGOT=0
00503    175*          DO 6045 ILIP=1,JOBS
00506    176*          IF(IWORK(ILIP).NE.0) IGOT=IGOT+1
00510    177*          IF(IGOT.GT.1) GO TO 6046
00512    178*     6045 CONTINUE
00514    179*          ISLT=ICAN(IBUL)
00515    180*          GO TO 8082
00516    181*     6046 CONTINUE
00516    182*     C
00516    183*     C    SELECT NODES TO ENTER THE SOLUTION SET.
00516    184*     C
00517    185*          CALL EVAL(IWORK,ISTER,ISLT,IQ,IA,IQD,IAD,JOBS,MACHS,NFIX,IU,IV,J,A
00517    186*         1T,MTSUM,MRATE,IDATE,JPA,JPB,ICAN,MAJB,IVALUE,JLIM)
00520    187*     8082 CONTINUE
00521    188*          DO 4582 IZIP=1,JOBS
00524    189*          DO 4583 IPIZ=1,JOBS
00527    190*          IF(ICAN(IZIP).EQ.IWORK(IPIZ)) GO TO 4586
00531    191*     4583 CONTINUE
00533    192*          GO TO 4582
00534    193*     4586 ICAN(IZIP)=0
00535    194*     4582 CONTINUE
00535    195*     C
00535    196*     C    PLACE OPERATIONS TO BE SCHEDULED INTO VECTOR IIN1
00535    197*     C
00537    198*          DO 6062 ITBB=1,JOBS
00542    199*          IF(IIN1(ITBB).EQ.0) GO TO 6063
00544    200*     6062 CONTINUE
00546    201*     6063 IIN1(ITBB)=ISLT
00547    202*          DO 3636 ITBB=1,JOBS
00552    203*          IWORK(ITBB)=0
00553    204*     3636 CONTINUE
00555    205*     6040 CONTINUE
00557    206*          GO TO 8713
00560    207*     5001 CONTINUE
00561    208*          DO 5262 ITBB=1,IBUM
00564    209*          IIN1(ITBB)=ISET(ITBB)
00565    210*     5262 CONTINUE
00567    211*     8713 CONTINUE
00570    212*          ISUM=0
00571    213*          DO 6088 ITBB=1,JOBS
00574    214*          IF(IIN1(ITBB).EQ.0) GO TO 6088
00576    215*          ISUM=ISUM+1
00577    216*     6088 CONTINUE
00601    217*          IF(IPRIN.EQ.1) GO TO 4044
00603    218*          WRITE(6,9023)
00605    219*     9023 FORMAT('  ','THE NODES SELECTED FOR N, ARE*)
00606    220*          WRITE(6,1)(IIN1(ITBB),ITBB=1,JOBS)
00606    221*     C
00606    222*     C    REARRANGE THE ARCS OF THE GRAPH
00606    223*     C
00614    224*     4044 CONTINUE
00615    225*          DO 79 IQB=1,JOBS
00620    226*          IF(IIN1(IQB).EQ.0) GO TO 79
00622    227*          IEE=IIN1(IQB)
00623    228*          IWW=IG(IEE)/100
00624    229*          IWWW=IG(IEE)-IWW*100
00625    230*          DO 80 IQV=1,JOBS
00630    231*          IF(J(IWWW,IQV).EQ.IIN1(IQB)) GO TO 80
00632    232*          IEX=J(IWWW,IQV)
00633    233*          IF(NFIX(IEX).EQ.1) GO TO 80
00635    234*          IBIF=0
00636    235*          DO 81 ITB=1,K
00641    236*          IF(IA(ITB).EQ.IIN1(IQB)) GO TO 82
00643    237*          GO TO 81
00644    238*       82 INEWT=IQ(ITB)
00645    239*          IBIF=1
00646    240*          GO TO 1082
00647    241*       81 CONTINUE
00651    242*     1082 CONTINUE
00652    243*          IF(IBIF.EQ.0) GO TO 3319
00654    244*          DO 85 ITB=1,K
00657    245*          IF(IQ(ITB).EQ.INEWT) GO TO 86
00661    246*          GO TO 85
00662    247*       86 IF(IA(ITB).EQ.IIN1(IQB)) GO TO 85
00664    248*          IQ(ITB)=IIN1(IQB)
00665    249*          IA(ITB)=J(IWWW,IQV)
00666    250*          GO TO 80
00667    251*       85 CONTINUE
```

```
00671   252*            GO TO 80
00672   253*         3319 CONTINUE
00673   254*            DO 3321 IUK=1,K
00676   255*            IF(IA(IUK),EQ.0) GO TO 3322
00700   256*            GO TO 3321
00701   257*         3322 IA(IUK)=J(IWWW,IQV)
00702   258*            IG(IUK)=TIN1(IQB)
00703   259*            GO TO 80
00704   260*         3321 CONTINUE
00706   261*           80 CONTINUE
00710   262*           79 CONTINUE
00712   263*            DO 6087 IGT=1,JOBS
00715   264*            IF(IIN1(IGT),EQ.0) GO TO 6087
00717   265*            IEX=TIN1(IGT)
00720   266*            NFIX(IEX)=1
00721   267*         6087 CONTINUE
00721   268*         C
00721   269*         C     UPDATE THE GRAPH.
00721   270*         C
00723   271*            CALL UPDATE(IA,IQ,K,IU,IV,MACHS,JOBS,J,IT,JM,NFIX)
00724   272*            IF(IIND.EQ.1) GO TO 9690
00724   273*         C
00724   274*         C     UPDATE THE LONGEST PATH.
00724   275*         C
00726   276*            IHOLDX=0
00727   277*            DO 4743 I=1,JM
00732   278*            IYIP=IU(I)-IV(I)
00733   279*            IF(IYIP.LE.0) GO TO 4744
00735   280*            IYIPP=IU(I)+IT(I)
00736   281*            GO TO 4745
00737   282*         4744 IYIPP=IV(I)+IT(I)
00740   283*         4745 IF(IYIPP.GT.IHOLDX) IHOLDX=IYIPP
00742   284*         4743 CONTINUE
00744   285*            ISTER=IHOLDX
00745   286*            IF(IPRIN.EQ.1) GO TO 4045
00747   287*            WRITE(6,27) ISTER
00752   288*           27 FORMAT(' ','THE NEW TERMINAL START TIME IS',I10)
00753   289*         4045 CONTINUE
00754   290*            RETURN
00755   291*            END



00101     1*            SUBROUTINE EVAL(IWORK,ISTER,ISLT,IQ,IA,IQD,IAD,JOBS,MACHS,NFIX,IU,
00101     2*           1IV,J,IT,MTSUM,MRATE,IDATE,JPA,JPB,ICAN,BJB,IVALUE,JLIM)
00101     3*         C
00101     4*         C
00101     5*         C
00101     6*         C     THIS ROUTINE TEMPORARILY ARRANGES THE ARCS OF THE GRAPH FOR
00101     7*         C     RESOLUTION OF THE CURRENT CONFLICT
00101     8*         C
00101     9*         C
00101    10*         C
00103    11*            DIMENSION IWORK(30),IQ(200),IA(200),IT(200),IQD(200),IAD(200),NFIX
00103    12*           1(200),ITIE(30),IUU(200),IVV(200),IU(200),IV(200),J(10,30),MTSUM(10
00103    13*           2),MRATE(10),JPA(30),JPB(30),IDATE(30),ICAN(30),IVALUE(30,11)
00104    14*            1 FORMAT(1X,11I7)
00105    15*              IGAPI=1
00106    16*              ITAX=9999999999
00107    17*              ITOT=0
00110    18*              IDLE=0
00111    19*              IPRIN=1
00112    20*              DO 47 IQQ=1,JOBS
00115    21*              ITIE(IQQ)=0
00116    22*           47 CONTINUE
00120    23*              JM=JOBS*MACHS
00121    24*              K=(JOBS-1)*MACHS
00122    25*              DO 50 IDU=1,K
00125    26*              IQD(IDU)=IQ(IDU)
00126    27*              IAD(IDU)=IA(IDU)
00127    28*           50 CONTINUE
00131    29*              DO 51 I=1,JM
00134    30*              IUU(I)=IU(I)
00135    31*              IVV(I)=IV(I)
00136    32*           51 CONTINUE
00140    33*              ICKK=99999
00141    34*              ISTIR=ISTER
00142    35*              IKEEP=0
00143    36*              ITE=0
00144    37*              DO 5000 IQQ=1,JOBS
```

```
00147    38*          IF(IPRIN.EQ.1) GO TO 4040
00151    39*          WRITE(6,1) IWORK(IOQ)
00154    40*     4040 CONTINUE
00155    41*          IF(IWORK(IOQ).EQ.0) GO TO 5000
00157    42*          DO 5001 IJIP=1,MACHS
00162    43*          DO 5002 IIIP=1,JOBS
00165    44*          IF(J(IJIP,IIIP).EQ.IWORK(IOQ)) GO TO 5010
00167    45*     5002 CONTINUE
00171    46*     5001 CONTINUE
00173    47*     5010 CONTINUE
00174    48*          ICLI=0
00175    49*          DO 5004 IJ=1,K
00200    50*          IF(IA(IJ).NE.IWORK(IOQ)) GO TO 5004
00202    51*          ICLI=1
00203    52*          IQ(IJ)=IQD(IJ)
00204    53*          IA(IJ)=IAD(IJ)
00205    54*          IBAS=IWORK(IOQ)
00206    55*     5004 CONTINUE
00210    56*          IF(ICLI.EQ.0) IBAS=IWORK(IOQ)
00212    57*          DO 5005 ILOW=1,JOBS
00215    58*          ITAK=J(IJIP,ILOW)
00216    59*          IF(NFIX(ITAK).EQ.1) GO TO 5005
00220    60*          IF(ITAK.EQ.IWORK(IOQ)) GO TO 5005
00222    61*          IF(ICLI.EQ.0) GO TO 4791
00224    62*          DO 5011 IJ=1,K
00227    63*          IF(IA(IJ).EQ.ITAK) GO TO 5012
00231    64*          GO TO 5011
00232    65*     5012 IQ(IJ)=IBAS
00233    66*          IA(IJ)=ITAK
00234    67*          GO TO 5005
00235    68*     5011 CONTINUE
00237    69*          GO TO 5005
00240    70*     4791 CONTINUE
00241    71*          DO 4692 IPIP=1,K
00244    72*          IF(IA(IPIP).EQ.0) GO TO 4693
00246    73*          GO TO 4692
00247    74*     4693 IQ(IPIP)=IBAS
00250    75*          IA(IPIP)=ITAK
00251    76*          GO TO 5005
00252    77*     4692 CONTINUE
00254    78*     5005 CONTINUE
00256    79*          CALL UPDATE(IA,IQ,K,IU,IV,MACHS,JOBS,J,IT,JM,NFIX)
00257    80*          CALL RESOLV(J,NFIX,IU,IV,JOBS,MACHS,ITOT,MTSUM,IJIP,IBAS,MRATE,IDA
00257    81*         1TE,JPA,JPB,IT,ICAN,IWORK,MBJB,IVALUE,JLIM)
00260    82*          IF(ITOT.LT.ITAX) GO TO 6287
00262    83*          GO TO 5208
00263    84*     6287 ITAX=ITOT
00264    85*          ISLT=IWORK(IOQ)
00265    86*          WRITE(6,1) ITAX
00270    87*     5208 DO 5209 IJ=1,K
00273    88*          IQ(IJ)=IQD(IJ)
00274    89*          IA(IJ)=IAD(IJ)
00275    90*     5209 CONTINUE
00277    91*          DO 7311 I=1,JM
00302    92*          IV(I)=IVV(I)
00303    93*          IU(I)=IUU(I)
00304    94*     7311 CONTINUE
00306    95*     5000 CONTINUE
00310    96*      959 CONTINUE
00311    97*          RETURN
00312    98*          END


00101     1*          SUBROUTINE UPDATE(IA,IQ,K,IU,IV,MACHS,JOBS,J,IT,JM,NFIX)
00101     2*     C
00101     3*     C
00101     4*     C
00101     5*     C    THIS ROUTINE UPDATES THE CURRENT GRAPH.
00101     6*     C
00101     7*     C
00101     8*     C
00103     9*          DIMENSION IA(200),IQ(200),IU(200),IV(200),IT(200),J(10,30),IUS(200
00103    10*         1),NFIX(200)
00104    11*           IPRIN=1
00105    12*         1 FORMAT(16I5)
00106    13*           IPIN=0
00107    14*       652 DO 653 I=1,JM
```

```
00112     15*          IUS(I)=0
00113     16*          IF(NFIX(I).EQ.1) IUS(I)=1
00115     17*      653 CONTINUE
00117     18*          L=0
00120     19*          IH=1
00121     20*          DO 800 I=1,JM
00124     21*          IF(NFIX(I).EQ.1) GO TO 1800
00126     22*          ICC=0
00127     23*          DO 801 IC=1,K
00132     24*          IF(IA(IC).EQ.I) ICC=1
00134     25*      801 CONTINUE
00136     26*          IF(ICC.EQ.0) GO TO 1801
00140     27*          GO TO 1800
00141     28*     1801 IF(I.EQ.IH) IUS(I)=1
00143     29*          IMIX=I-1
00144     30*          IF(IMIX.EQ.0) GO TO 1803
00146     31*          IF(IUS(IMIX).EQ.1) IUS(I)=1
00150     32*     1803 CONTINUE
00151     33*          IV(I)=0
00152     34*          IP=I+1
00153     35*          IF(IP.EQ.IH) GO TO 1800
00155     36*          IU(IP)=IU(I)+IT(I)
00156     37*     1800 IF(I.EQ.IH) IH=IH+MACHS
00160     38*      800 CONTINUE
00162     39*          IF(IPRIN.EQ.1) GO TO 4040
00164     40*          WRITE(6,1)(IUS(I),I=1,JM)
00172     41*     4040 CONTINUE
00173     42*      654 L=L+1
00174     43*          ICZ=L
00175     44*          NNN=((JOBS-1)*MACHS)+L
00176     45*      670 CONTINUE
00177     46*          IF(NFIX(ICZ).EQ.1) GO TO 782
00201     47*          IHH=ICZ-1
00202     48*          IKEEP=0
00203     49*          IHA=0
00204     50*          IVIS=0
00205     51*          IVIT=0
00206     52*      658 DO 660 I=1,K
00211     53*          IF(IA(I).EQ.ICZ) GO TO 661
00213     54*          GO TO 660
00214     55*      661 CONTINUE
00215     56*          IF(IHA.EQ.1) GO TO 660
00217     57*          IEE=IQ(I)
00220     58*          IF(IUS(IEE).EQ.0) GO TO 780
00222     59*      781 GO TO 662
00223     60*      780 IHA=1
00224     61*          ISAV=0
00225     62*          GO TO 660
00226     63*      662 IJI=IU(IEE)-IV(IEE)
00227     64*          IF(IJI.LE.0) GO TO 663
00231     65*          ISAV=IU(IEE)+IT(IEE)
00232     66*          GO TO 664
00233     67*      663 ISAV=IV(IEE)+IT(IEE)
00234     68*      664 IF(ISAV.LT.IKEEP) GO TO 660
00236     69*          IKEEP=ISAV
00237     70*          IF(IHA.EQ.1) GO TO 660
00241     71*          IVIS=IA(I)
00242     72*          IVIT=IQ(I)
00243     73*      660 CONTINUE
00245     74*          IF(IVIS.EQ.0) GO TO 1804
00247     75*          GO TO 1805
00250     76*     1804 IF(IUS(ICZ).EQ.1) GO TO 1805
00252     77*          IF(IHH.EQ.0) GO TO 1805
00254     78*          IF(IUS(IHH).EQ.1.AND.IHA.EQ.0) IUS(ICZ)=1
00256     79*     1805 CONTINUE
00256     80*     C
00256     81*     C    UPDATE SUBSEQUENT AND CURRENT NODE START TIMES.
00256     82*     C
00257     83*          IF(IHA.GT.0) GO TO 782
00261     84*          IF(IVIS.EQ.0) GO TO 782
00263     85*          ICZZ=MACHS-L
00264     86*          IICZ=ICZZ+IVIS
00265     87*          JIO=IV(IVIS)
00266     88*          IF(IKEEP.GT.JIO) GO TO 1910
00270     89*          GO TO 1920
00271     90*     1910 IV(IVIS)=IKEEP
00272     91*     1920 IUS(IVIS)=1
00273     92*          DO 708 I=IVIS,IICZ
00276     93*          IF(I.EQ.IICZ) GO TO 708
00300     94*          ICOL=I+1
```

```
00301     95*           IMUM=IU(I)-IV(I)
00302     96*           IF(IMUM.GT.0) GO TO 710
00304     97*           GO TO 711
00305     98*       710 IU(ICOL)=IU(I)+IT(I)
00306     99*           GO TO 708
00307    100*       711 IU(ICOL)=IV(I)+IT(I)
00310    101*       708 CONTINUE
00312    102*           IF(IFIN.EQ.1) GO TO 767
00314    103*       762 IF(ICZ.EQ.NNN) GO TO 750
00316    104*           ICZ=ICZ+MACHS
00317    105*           GO TO 670
00320    106*       750 IF(L.EQ.MACHS) GO TO 760
00322    107*           GO TO 654
00323    108*       760 IFIN=1
00324    109*           IF(IPRIN.EQ.1) GO TO 4041
00326    110*           WRITE(6,1)(IUS(I),I=1,JM)
00334    111*      4041 CONTINUE
00335    112*           L=0
00336    113*       761 L=L+1
00337    114*           NNN=((JOBS-1)*MACHS)+L
00340    115*       762 ICZ=L
00341    116*       763 IF(IUS(ICZ).EQ.0) GO TO 670
00343    117*       767 IF(ICZ.EQ.NNN) GO TO 764
00345    118*           ICZ=ICZ+MACHS
00346    119*           GO TO 763
00347    120*       764 IF(L.EQ.MACHS) GO TO 765
00351    121*           GO TO 761
00352    122*       765 CONTINUE
00353    123*           IF(IPRIN.EQ.1) GO TO 4042
00355    124*           WRITE(6,1)(IUS(I),I=1,JM)
00363    125*      4042 CONTINUE
00364    126*           DO 3342 I=1,JM
00367    127*           IF(IUS(I).NE.0) GO TO 3342
00371    128*           ICZ=I
00372    129*           NNN=I
00373    130*           GO TO 670
00374    131*      3342 CONTINUE
00376    132*           RETURN
00377    133*           END
```

```
00101      1*           SUBROUTINE RESOLV(J,NFIX,IU,IV,JOBS,MACHS,ITOT,MTSUM,IJIP,IBAS,MRA
00101      2*          1TE,IDATE,JPA,JPB,IT,ICAN,IWORK,MBJB,IVALUE,JLIM)
00103      3*           DIMENSION J(10,30),NFIX(200),IU(200),IV(200),MTSUM(10),MRATE(10),I
00103      4*          1DATE(30),JPA(30),JPB(30),IT(200),ICAN(30),IWORK(30),IVALUE(30,11)
00103      5*      C
00103      6*      C
00103      7*      C    THIS SUBROUTINE COMPUTES A LOWER BOUND ON MACHINE IDLE COST AND JOB
00103      8*      C    PENALTY COST IF CONFLICT IS RESOLVED IN FAVOR OF OPERATION 'IBAS'
00103      9*      C
00103     10*      C              MBJB    0   CONFLICT RESOLUTION BASED ON COST.
00103     11*      C                      1   CONFLICT RESOLUTION BASED ON COMPOSITE MACHINE
00103     12*      C                          BASED AND JOB BASED BOUND.
00103     13*      C              JLIM    0   NO LIMIT ON PENALTY COST
00103     14*      C                      1   PENALTY COST LIMITED BY JOB VALUE
00103     15*      C
00104     16*           LIMV=2
00105     17*           MAK=MACHS+1
00106     18*           MB=0
00107     19*           DO 9390 IMX=1,MACHS
00112     20*           ITSUM=0
00113     21*           IF(IMX.EQ.IJIP) GO TO 9410
00115     22*           ISTART=0
00116     23*           IIND=1
00117     24*           DO 9370 IJX=1,JOBS
00122     25*           IPIC=J(IMX,IJX)
00123     26*           IF(NFIX(IPIC).EQ.1) GO TO 9370
00125     27*           IF(IU(IPIC).GT.IV(IPIC)) GO TO 9341
00127     28*           ISX=IV(IPIC)
00130     29*           GO TO 9350
00131     30*      9341 ISX=IU(IPIC)
00132     31*      9350 IF(IIND.EQ.1) GO TO 9355
00134     32*           GO TO 9360
00135     33*      9355 ISTART=ISX
00136     34*           IIND=0
00137     35*      9360 IF(ISX.GE.ISTART) GO TO 9365
00141     36*           ISTART=ISX
00142     37*      9365 ITSUM=ITSUM+IT(IPIC)
00143     38*      9370 CONTINUE
```

```
00145     39*           MTIMF=ISTART+ITSUM
00146     40*           GO TO 9380
00147     41*      9410 DO 9440 IJX=1,JOBS
00152     42*           IPIC=J(IMX,IJX)
00153     43*           IF(NFIX(IPIC).EQ.1) GO TO 9440
00155     44*           IF(IPIC.EQ.IBAS) GO TO 9430
00157     45*           ITSUM=ITSUM+IT(IPIC)
00160     46*           GO TO 9440
00161     47*      9430 IF(IU(IBAS).GT.IV(IBAS)) GO TO 9435
00163     48*           ICOMP=IV(IBAS)+IT(IBAS)
00164     49*           GO TO 9440
00165     50*      9435 ICOMP=IU(IBAS)+IT(IBAS)
00166     51*      9440 CONTINUE
00170     52*           MTIME=ICOMP+ITSUM
00171     53*      9380 IF(MTIME.GT.MB) MB=MTIME
00173     54*      9390 CONTINUE
00175     55*           MBOUND=0
00176     56*           IF(MRJB.EQ.1) GO TO 9451
00200     57*           DO 9450 K=1,MACHS
00203     58*           MBOUND=MBOUND+((MRATE(K))*(MB-MTSUM(K)))
00204     59*      9450 CONTINUE
00206     60*      9451 JB=0
00207     61*           JBOUND=0
00210     62*           IX=1
00211     63*           KX=MACHS
00212     64*           DO 9490 L=1,JOBS
00215     65*           IF(IU(KX).GT.IV(KX)) GO TO 9465
00217     66*           JTSUM=IV(KX)+IT(KX)
00220     67*           GO TO 9470
00221     68*      9465 JTSUM=IU(KX)+IT(KX)
00222     69*      9470 ITR=JTSUM-IDATE(L)
00223     70*           IF(MRJB.EQ.1) GO TO 9481
00225     71*           IF(ITR.GT.0) GO TO 9485
00227     72*           ITARD=0
00230     73*           GO TO 9486
00231     74*      9485 ITARD=ITR
00232     75*      9486 JPEN=JPA(L)*ITARD+JPB(L)*(ITARD**2)
00233     76*           IF(JLIM.EQ.0) GO TO 9820
00235     77*           LIMT=LIMV*IVALUE(L,MAK)
00236     78*           IF(LIMT.LT.JPEN) JPEN=LIMT
00240     79*      9820 JBOUND=JBOUND+JPEN
00241     80*           GO TO 9488
00242     81*      9481 ITR=JTSUM
00243     82*           IF(ITR.GT.JB) JB=ITR
00245     83*      9488 IX=IX+MACHS
00246     84*           KX=KX+MACHS
00247     85*      9490 CONTINUE
00251     86*           ITOT=MBOUND+JBOUND
00252     87*           IF(MRJB.EQ.0) GO TO 9493
00254     88*      9491 IF(JB.GT.MB) GO TO 9492
00256     89*           ITOT=MB
00257     90*           GO TO 9493
00260     91*      9492 ITOT=JB
00261     92*      9493 CONTINUE
00262     93*           RETURN
00263     94*           END
```

BIBLIOGRAPHY

1. Ashour, S., "A Decomposition Approach for the Machine Scheduling Problem," The International Journal of Production Research, Vol. 6, No. 2, 1967, pp. 109-122.

2. Ashour, S., "An Experimental Investigation and Comparative Evaluation of Flow-Shop Scheduling Techniques," Operations Research, Vol. 18, No. 3, 1970, pp. 541-549.

3. Ashour, S., Sequencing Theory, Springer-Verlag, Berlin, 1972.

4. Ashour, S., and S. R. Hiremath, "A Branch-and-Bound Approach to the Job-Shop Scheduling Problem," The International Journal of Production Research, Vol. 11, No. 1, 1973, pp. 47-58.

5. Ashour, S., and R. G. Parker, "A Precedence Graph Algorithm for the Shop Scheduling Problem," Operational Research Quarterly, Vol. 22, No. 2, 1971, pp. 165-175.

6. Ashour, S., and R. G. Parker, "An Out-of-Kilter Approach for Machine Sequencing Problems," presented at the Symposium on Theory of Scheduling and its Application, North Carolina State University, Raleigh, N. C., May 15-17, 1972.

7. Ashour, S., and M. N. Quraishi, "Investigation of Various Bounding Procedures for Production Scheduling Problems," The International Journal of Production Research, Vol. 7, No. 3, 1969, pp. 249-252.

8. Balas, E., "Discrete Programming by the Filter Method," Operations Research, Vol. 15, No. 5, 1967, pp. 915-957.

9. Balas, E., "Machine Sequencing Via Disjunctive Graphs: An Implicit Enumeration Algorithm," Operations Research, Vol. 17, No. 6, 1969, pp. 941-957.

10. Beenhakker, H. L., "Mathematical Analysis of Facility-Commodity Scheduling Problems," International Journal of Production Research, Vol. 2, No. 4, 1963, pp. 313-321.

11. Brooks, G. H., and C. R. White, "An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem," The Journal of Industrial Engineering, Vol XVI, No. 1, 1965, pp. 34-40.

12. Brown, A. P. G., and Z. A. Lomnicki, "Some Applications of the 'Branch-and-Bound' Algorithm to the Machine Scheduling Problem,"

Operational Research Quarterly, Vol. 17, No. 2, 1966, pp. 173-186.

13. Charlton, J. M., and C. C. Death, "A Generalized Machine-Scheduling Algorithm," Operational Research Quarterly, Vol. 21, No. 1, 1970, pp. 127-134.

14. Conway, R. W., "Priority Despatching and Work-in-Process Inventory in a Job Shop," The Journal of Industrial Engineering, Vol. XVI, No. 2, 1965, pp. 123-130.

15. Conway, R. W., "Priority Dispatching and Job Lateness in a Job Shop," The Journal of Industrial Engineering, Vol. XVI, No. 4, 1965, pp. 228-237.

16. Conway, R. W., B. M. Johnson, and W. L. Maxwell, "An Experimental Investigation of Priority Dispatching," The Journal of Industrial Engineering, Vol. XI, No. 3, 1960, pp. 221-229.

17. Conway, R. W., W. L. Maxwell, and L. W. Miller, Theory of Scheduling, Addison-Wesley Co., Reading, Massachusetts, 1967.

18. Day, J. E., and M. P. Hottenstein, "Review of Sequencing Research," Naval Research Logistics Quarterly, Vol. 17, No. 1, 1970, pp. 11-39.

19. Deane, R. H., and C. L. Moodie, "A Dispatching Methodology for Balancing Workload Assigrments in a Job Shop Production Facility," AIIE Transactions, Vol. 4, No. 4, 1972, pp. 277-283.

20. Dudek, R. A., and O. F. Teuton, Jr., "Development of M-Stage Decision Rule for Scheduling n Jobs Through M Machines," Operations Research, Vol. 12, No. 3, 1964, pp. 471-497.

21. Eilon, S., and J. R. King, Industrial Scheduling Abstracts (1950-1966), Oliver and Boyd, Edinburgh and London, 1967.

22. Elmaghraby, S. E., The Design of Production Systems, Reinhold Publishing Corporation, New York, 1966.

23. Elmaghraby, S. E., "The One Machine Sequencing Problem with Delay Costs," The Journal of Industrial Engineering, Vol. XIX, No. 2, 1968, pp. 105-108.

24. Elmaghraby, S. E., "The Machine Sequencing Problem-Review and Extensions," Naval Research Logistics Quarterly, Vol. 15, No. 2, 1968, pp. 205-232.

25. Emmons, H., "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness," Operations Research, Vol. 17, No. 4, 1969, pp. 701-715.

26. Fisher, H., and G. L. Thompson, "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules," Chapter 15 in reference 53.

27. Florian, M., P. Trepant, and G. McMahon, "An Implicit Enumeration Algorithm for the Machine Sequencing Problem," Management Science, Vol. 17, No. 12, 1971, pp. B783-B792.

28. Gapp, W., P. S. Mankekar, and L. G. Mitten, "Sequencing Operations to Minimize In-Process Inventory Costs," Management Science, Vol. 11, No. 3, 1965, pp. 476-484.

29. Gere, W. S., Jr., "Heuristics in Job Shop Scheduling," Management Science, Vol. 13, No. 3, 1966, pp. 167-190.

30. Giffler, B., and G. L. Thompson, "Algorithms for Solving Production Scheduling Problems," Operations Research, Vol. 8, No. 4, 1960, pp. 487-503.

31. Giglio, R. J., and H. M. Wagner, "Approximate Solutions to the Three Machine Scheduling Problem," Operations Research, Vol. 12, No. 2, 1964, pp. 305-324.

32. Greenberg, H. H., "A Branch and Bound Solution to the General Scheduling Problem," Operations Research, Vol. 16, No. 2, 1968, pp. 353-361.

33. Gupta, J. N. D., "Economic Aspects of Scheduling Theory," Ph.D. Thesis, Texas Tech University, Lubbock, Texas, 1969.

34. Gupta, J. N. D., "M-Stage Scheduling Problem - A Critical Appraisal," International Journal of Production Research, Vol. 9, No. 2, 1971, pp. 267-281.

35. Gupta, J. N. D., "Economic Aspects of Production Scheduling Systems," Journal of the Operations Research Society of Japan, Vol. 13, No. 4, 1971, pp. 169-193.

36. Gupta, J. N. D., "Optimality Criteria for Flowshop Schedules," AIIE Transactions, Vol. III, No. 3, 1971, pp. 199-205.

37. Harvey, W. S., T. A. J. Nicholson, R. D. Pullen, and D. P. Quas, "The Optimization of Paper Machine Scheduling," Operational Research Quarterly, Vol. 20, No. 2, 1969, pp. 237-245.

38. Held, M., and R. M. Karp, "A Dynamic Programming Approach to Sequencing Problems," Journal of SIAM, Vol. 10, No. 1, 1962, pp. 196-210.

39. Heller, J., "Some Numerical Experiments for an M X J Flow Shop and Its Decision Theoretical Aspects," Operations Research, Vol. 8, No. 2, 1960, pp. 178-184.

40. Holt, C., "Priority Rules for Minimizing the Cost of Queues in Machine Scheduling," Chapter 6 in reference 53.

41. Ignall, E., and L. Schrage, "Application of the Branch and Bound Technique to Some Flow-Shop Scheduling Problems," Operations Research, Vol. 13, No. 3, 1965, pp. 400-412.

42. Johnson, S. M., "Optimal Two-and Three-Stage Production Schedules with Setup Times Included," Naval Research Logistics Quarterly, Vol. 1, No. 1, 1954, pp. 61-68.

43. Karush, W., "A Counterexample to a Proposed Algorithm for Optimal Sequencing of Jobs," Operations Research, Vol. 13, No. 2, 1965, pp. 323-325.

44. Lawler, E. L., "On Scheduling Problems With Deferral Costs," Management Science, Vol. 11, No. 2, 1964, pp. 280-288.

45. Lawler, E. L., and J. M. Moore, "A Functional Equation and Its Application to Resource Allocation and Sequencing Problems," Management Science, Vol. 16, No. 1, 1969, pp. 77-84.

46. LeGrande, E., "The Development of a Factory Simulation System Using Actual Operating Data," Management Technology, Vol. 13, No. 1, 1963, pp. 1-19.

47. Ligtenberg, E., "Minimal Cost Sequencing of n Grouped and Ordered Jobs on m Machines," The Journal of Industrial Engineering, Vol. XVII, No. 4, 1966, pp. 217-223.

48. Manne, A. S., "On the Job-Shop Scheduling Problem," Operations Research, Vol. 8, No. 2, 1960, pp. 219-223.

49. McMahon, G. B. and P. G. Burton, "Flow-Shop Scheduling with the Branch-and-Bound Method," Operations Research, Vol. 15, No. 3, 1967, pp. 473-481.

50. McNaughton, R., "Scheduling with Deadlines and Loss Functions," Management Science, Vol. 6, No. 1, 1959, pp. 1-12.

51. Mellor, P., "A Review of Job Shop Scheduling," Operational Research Quarterly, Vol. 17, No. 2, 1966, pp. 161-171.

52. Moore, J. M., "An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," Management Science, Vol. 15, No. 1, 1968, pp. 102-109.

53. Muth, J. F., and G. L. Thompson, eds., Industrial Scheduling, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1963.

54. Ostle, Bernard, Statistics in Research, Iowa State University Press, Ames, Iowa, 1963.

55. Root, J. G., "Scheduling with Deadlines and Loss Functions on K Parallel Machines," Management Science, Vol. 11, No. 3, 1965, pp. 460-475.

56. Shwimer, J., "On the N-Job, One-Machine, Sequence-Independent Scheduling Problem with Tardiness Penalties: A Branch and Bound Solution," Management Science, Vol. 18, No. 6, 1972, pp. B301-B313.

57. Smith, R. D., and R. A. Dudek, "A General Algorithm for Solution of the n-Job, M-Machine Sequencing Problem of the Flow Shop," Operations Research, Vol. 15, No. 1, 1967, pp. 71-82; also, "Errata," Operations Research, Vol. 17, No. 4, 1969, p. 756.

58. Smith, W. E., "Various Optimizers for Single Stage Production," Naval Research Logistics Quarterly, Vol. 3, 1956, pp. 59-66.

59. Wagner, H. M., "An Integer Linear-Programming Model for Machine Scheduling," Naval Research Logistics Quarterly, Vol. 6, No. 2, 1959, pp. 131-140.